FBISE

COMPUTER SC.

MODEL PAPERS & GUESS PAPERS

Faderal Board Islamabad

Presented by:

Urdu Books Whatsapp Group

STUDY GROUP

10TH CLASS

0333-8033313	0343-7008883	0306-7163117
راؤایاز	پاکستان زندہ باد	محمد سلمان سلیم

GUESS PAPER & MODEL PAPER # 5

Based on Chapter # 05 (Reduced Syllabus)

LOOP CONTROL STRUCTURE

SECTION-A (Marks 12)

Time allowed: 20 Minutes **Marks: 12**

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. **Which structure enables the programmer to execute a set of instructions repeatedly until a particular condition is met?**
A. selection B. sequence C. choice D. loop

ii. **Which of the following is also called counter loop?**
A. do-while B. while C. for D. if-else

iii. **Which of the following ends a multiple-statement while loop?**
A. right bracket B. right brace C. semicolon D. colon

iv. **Which loop is used to execute a set of statements repeatedly for a fixed number of times?**
A. for loop B. do while loop
C. while loop D. nested loop

v. **Which loop is used when it is required to execute the loop at least once?**
A. for loop B. do while loop
C. while loop D. nested loop

vi. **Which loop is preferred to use when the number of times the loop will execute is not known in advance?**
A. for loop B. do while loop
C. while loop D. nested loop

vii. **In which loop condition is placed at the end?**
A. for loop B. do while loop
C. while loop D. nested loop

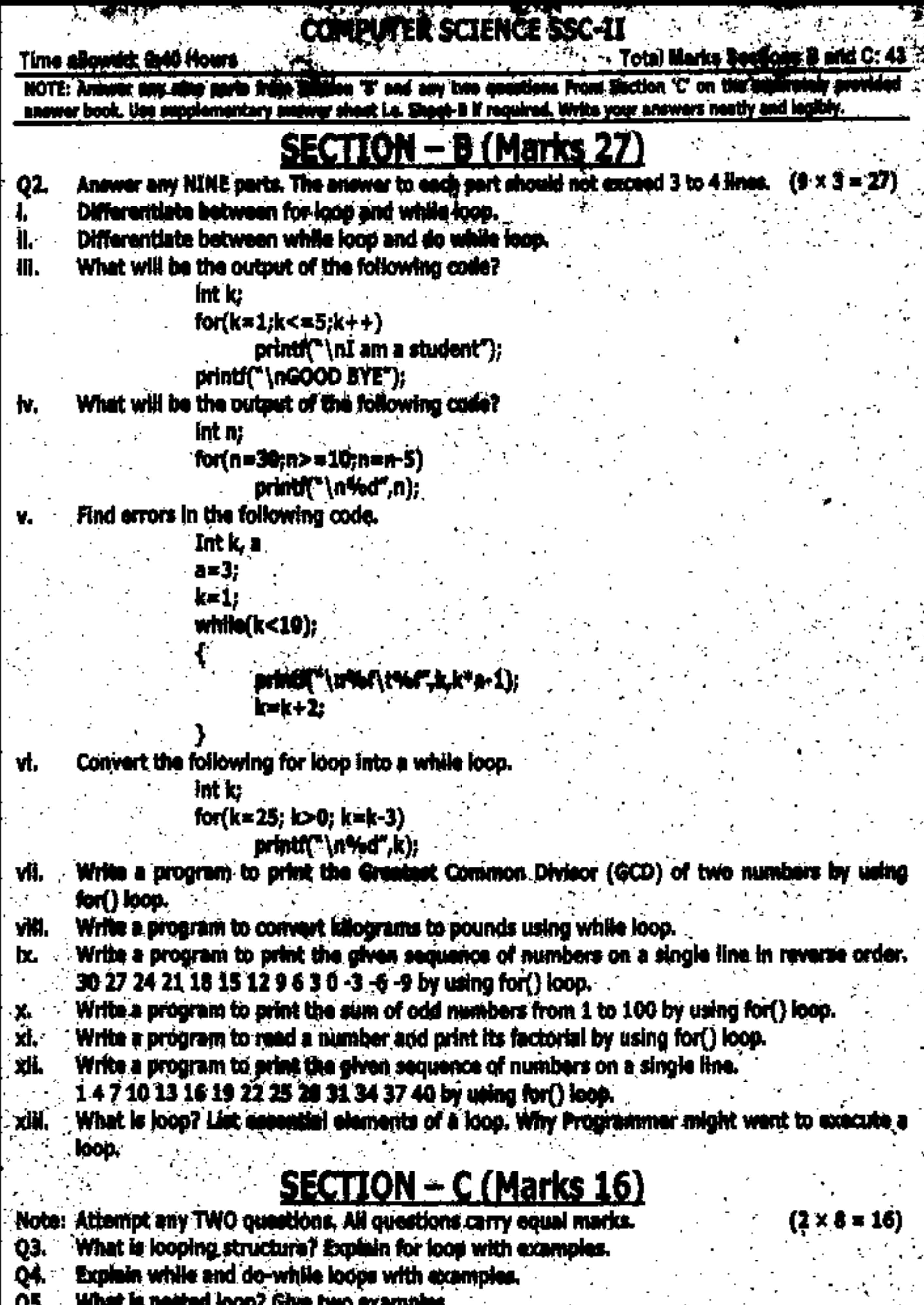
viii. **Which statement is used to exit from a loop as soon as certain condition is met?**
A. break B. continue C. if statement D. default

ix. **What is a pass through a loop called?**
A. execution B. iteration
C. enumeration D. culmination

x. **Which of the following should be placed at the beginning and the end of a for loop if it consists of more than one statement?**
A. () B. < > C. [] D. {}

xi. **Which looping process checks the test condition at the end of the loop?**
A. for loop B. do while loop
C. while loop D. None of these

xii. **Which looping process is best used when the number of iterations is known?**
A. for loop B. do while loop



COMPUTER SCIENCE SSC-II

Time allowed: 2:40 Hours — Total Marks Sections B and C: 43

NOTE: Answer any nine parts from Section 'B' and any two questions from Section 'C' on the separately provided answer book. Use supplementary answer sheet i.e. Sheet-B if required. Write your answers neatly and legibly.

SECTION – B (Marks 27)

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Differentiate between for loop and while loop.

ii. Differentiate between while loop and do while loop.

iii. What will be the output of the following code?

```
int k;
for(k=1;k<=5;k++)
        printf("\nI am a student");
printf("\nGOOD BYE");
```

iv. What will be the output of the following code?

```
int n;
for(n=30;n>=10;n=n-5)
        printf("\n%d",n);
```

v. Find errors in the following code.

```
Int k, a
a=3;
k=1;
while(k<10);
{
        printf("\n%f\t%f",k,k*a-1);
        k=k+2;
}
```

vi. Convert the following for loop into a while loop.

```
int k;
for(k=25; k>0; k=k-3)
        printf("\n%d",k);
```

vii. Write a program to print the Greatest Common Divisor (GCD) of two numbers by using for() loop.

viii. Write a program to convert kilograms to pounds using while loop.

ix. Write a program to print the given sequence of numbers on a single line in reverse order. 30 27 24 21 18 15 12 9 6 3 0 -3 -6 -9 by using for() loop.

x. Write a program to print the sum of odd numbers from 1 to 100 by using for() loop.

xi. Write a program to read a number and print its factorial by using for() loop.

xii. Write a program to print the given sequence of numbers on a single line.
1 4 7 10 13 16 19 22 25 28 31 34 37 40 by using for() loop.

xiii. What is loop? List essential elements of a loop. Why Programmer might want to execute a loop.

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. What is looping structure? Explain for loop with examples.

Q4. Explain while and do-while loops with examples.

Q5. What is nested loop? Give two examples.

ختم نبوت ﷺ زندہ باد **عظمت صحابہ زندہ باد**

السلام علیکم ورحمۃ اللہ وبرکاتہ:

معزز ممبران: آپ کا وٹس ایپ گروپ ایڈمن "**اردو بکس**" آپ سے مخاطب ہے۔

آپ تمام ممبران سے گزارش ہے کہ:

❖ گروپ میں صرف PDF کتب پوسٹ کی جاتی ہیں لہذا کتب کے متعلق اپنے کمنٹس / ریویوز ضرور دیں۔ گروپ میں بغیر ایڈمن کی اجازت کے کسی بھی قسم کی (اسلامی و غیر اسلامی، اخلاقی، تحریری) پوسٹ کرنا سختی سے منع ہے۔

❖ گروپ میں معزز، پڑھے لکھے، سلجھے ہوئے ممبرز موجود ہیں اخلاقیات کی پابندی کریں اور گروپ رولز کو فالو کریں بصورت دیگر معزز ممبرز کی بہتری کی خاطر ریموو کر دیا جائے گا۔

❖ کوئی بھی ممبر کسی بھی ممبر کو انباکس میں میسیج، مس کال، کال نہیں کرے گا۔ رپورٹ پر فوری ریموو کرکے کاروائی عمل میں لائے گی۔

❖ ہمارے کسی بھی گروپ میں سیاسی و فرقہ واریت کی بحث کی قطعاً کوئی گنجائش نہیں ہے۔

❖ اگر کسی کو بھی گروپ کے متعلق کسی قسم کی شکایت یا تجویز کی صورت میں ایڈمن سے رابطہ کیجئے۔

❖ سب سے اہم بات:

گروپ میں کسی بھی قادیانی، مرزائی، احمدی، گستاخِ رسول، گستاخِ امہات المؤمنین، گستاخِ صحابہ و خلفائے راشدین حضرت ابو بکر صدیق، حضرت عمر فاروق، حضرت عثمان غنی، حضرت علی المرتضیٰ، حضرت حسنین کریمین رضوان اللہ تعالی اجمعین، گستاخ اہلبیت یا ایسے غیر مسلم جو اسلام اور پاکستان کے خلاف پراپیگنڈا میں مصروف ہیں یا ان کے روحانی وذہنی سپورٹرز کے لئے کوئی گنجائش نہیں ہے لہذا ایسے اشخاص بالکل بھی گروپ جوائن کرنے کی زحمت نہ کریں۔ معلوم ہونے پر فوراً ریموو کر دیا جائے گا۔

❖ تمام کتب انٹرنیٹ سے تلاش / ڈاؤنلوڈ کرکے **فری آف کاسٹ** وٹس ایپ گروپ میں شیئر کی جاتی ہیں۔ جو کتاب نہیں ملتی اس کے لئے معذرت کر لی جاتی ہے۔ جس میں محنت بھی صَرف ہوتی ہے لیکن ہمیں آپ سے صرف دعاؤں کی درخواست ہے۔

❖ [redacted]

لیڈیز کے لئے الگ گروپ کی سہولت موجود ہے جس کے لئے ویریفکیشن ضروری ہے۔

❖ رابطہ کریں اور جواب کا انتظار فرمائیں۔ برائے مہربانی اخلاقیات کا خیال رکھتے ہوئے موبائل پر کال یا ایم ایس کرنے کی کوشش ہرگز نہ کریں۔ ورنہ گروپس سے تو ریموو کیا ہی جائے گا بلاک بھی کیا جائے گا۔

[redacted]

0306-7163117	0343-7008883	0333-8033313
محمد سلمان سلیم	پاکستان زندہ باد	راؤ اناز

[redacted]

SOLUTION OF GUESS PAPER & MODEL PAPER # 5 (Reduced Syllabus)

SECTION- A (MCQs)

i. D	ii. C	iii. B	iv. A	v. B	vi. C
vii. B	viii. A	ix. B	x. D	xi. B	xii. A

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Differentiate between for loop and while loop.

Ans:

For loop:

The for statement is used to executed a set of statements repeatedly for a fixed number of times in a program.

While loop:

The while statement is used to implement repetition structure in a program when the number of iterations is not known in advance and the repetition continues until some condition remains true.

ii. Differentiate between while loop and do while loop.

Ans: Difference Between While() And Do-While() Loops:

	WHILE() loop	DO-WHILE() loop
1.	While() loop is pre-tested loop.	Do-While loop is post-tested loop.

2.	The Syntax or general form of while() loop is:
	while(condition)
	{
	statements; //body of loop
	}
		The Syntax or general form of do-while() loop is:
		do{
		.
		statements; // body of loop.
		.
		} while(Condition);

3.	In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
4.	The iterations do not occur if, the condition at the first iteration appears false.	The iteration occurs at least once even if the condition is false at the first iteration.

iii. What will be the output of the following code?

```
int k;
for(k=1;k<=5;k++)
        printf("\nI am a student");
printf("\nGOOD BYE");
```

Ans:

```
I am a student
I am a student
I am a student
I am a student
```

iv. What will be the output of the following code?

```
            Int n;
            for(n=30;n>=10;n=n-5)
                  printf("\n%d",n);
```

Ans:

```
30
25
20
15
10
```

v. Find errors in the following code.

```
            Int k, a
            a=3;
            k=1;
            while(k<10);
            {
                  printf("\n%f\t%f",k,k*a-1);
                  k=k+2;
            }
```

Ans:

```
      Int a,k;
      a=3;
      k=1;
      while(k<10)
      {
            printf("\n%d\t%d",k,k*a-1);
            k=k+2;
      }
```

Output:

```
1     2
3     8
5     14
7     20
9     26
```

vi. Convert the following for loop into a while loop.

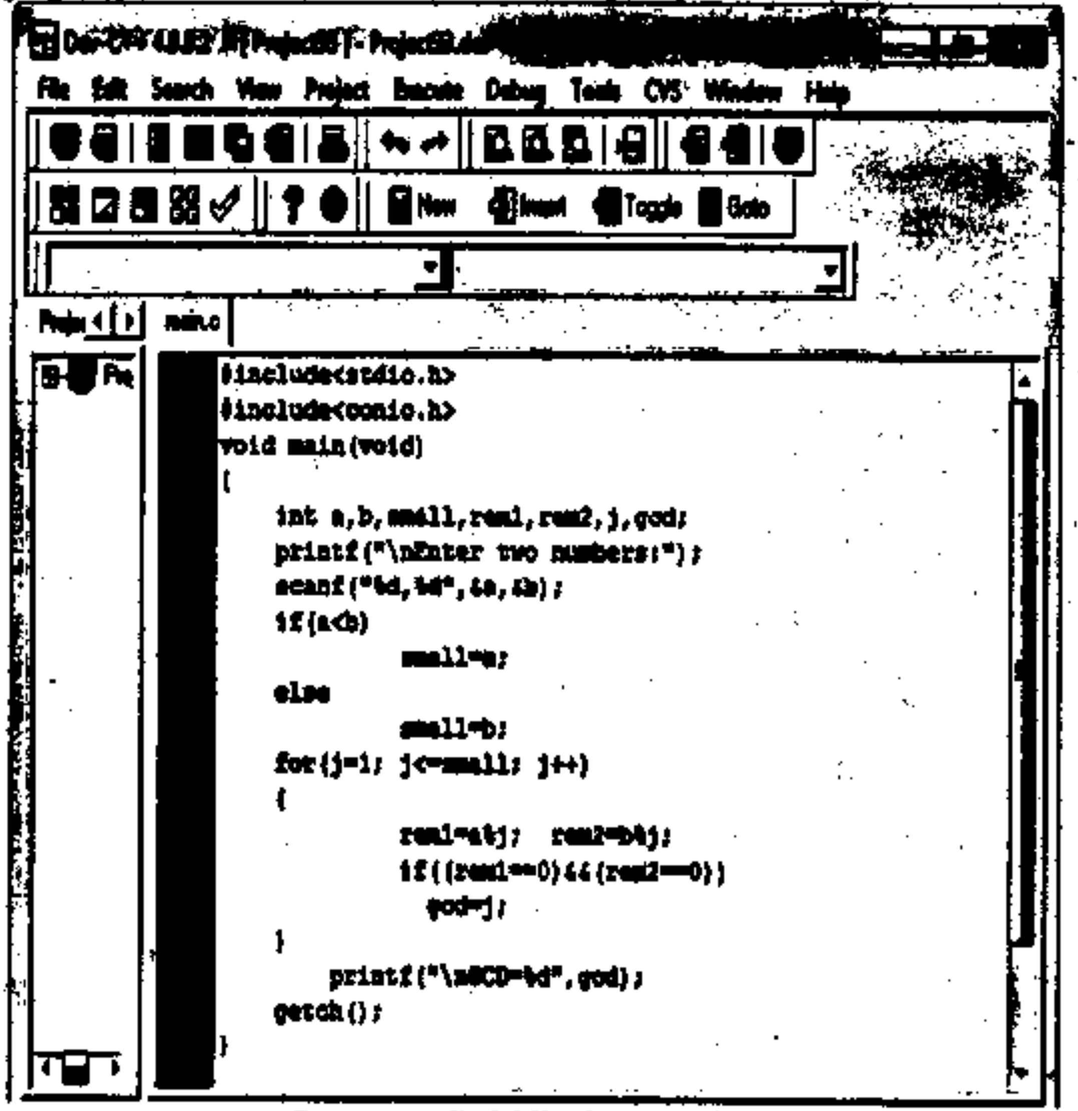
```
            Int k;
            for(k=25; k>0; k=k-3)
                  printf("\n%d",k);
```

Ans:

```
      int k;
      k=25;
      while(k>0)
      {
            printf("\n%d",k);
            k=k-3;
      }
```

vii. **Write a program to print the Greatest Common Divisor (GCD) of two numbers by using for() loop.**

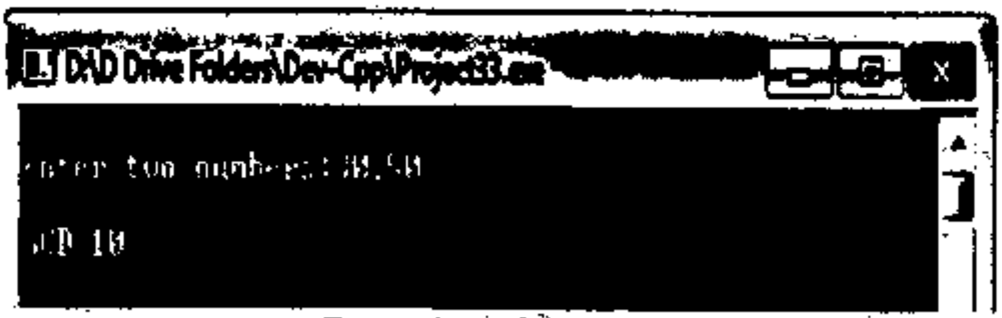
Ans: **Program to print the Greatest Common Divisor (GCD) of two numbers.**


File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int a,b,small,rem1,rem2,j,gcd;
    printf("\nEnter two numbers:");
    scanf("%d,%d",&a,&b);
    if(a<b)
            small=a;
    else
            small=b;
    for(j=1; j<=small; j++)
    {
            rem1=a%j;  rem2=b%j;
            if((rem1==0)&&(rem2==0))
              gcd=j;
    }
        printf("\nGCD=%d",gcd);
    getch();
}
```

Program to find GCD of two numbers

- In this program a and b are two variables whose GCD is required. After reading the two numbers from the keyboard, **if-else** statement will determine the smaller of the two numbers and assign it to the variable **small**.
- The GCD is in the range of 1 to the smaller number. The two variables **rem1** and **rem2** are used to determine whether the numbers a and b are exactly divisible by j by using the remainder operator.
- The greatest value of j that exactly divides both variables without any remainder is assigned to the variable **gcd**. Execution of this program is shown in Figure.

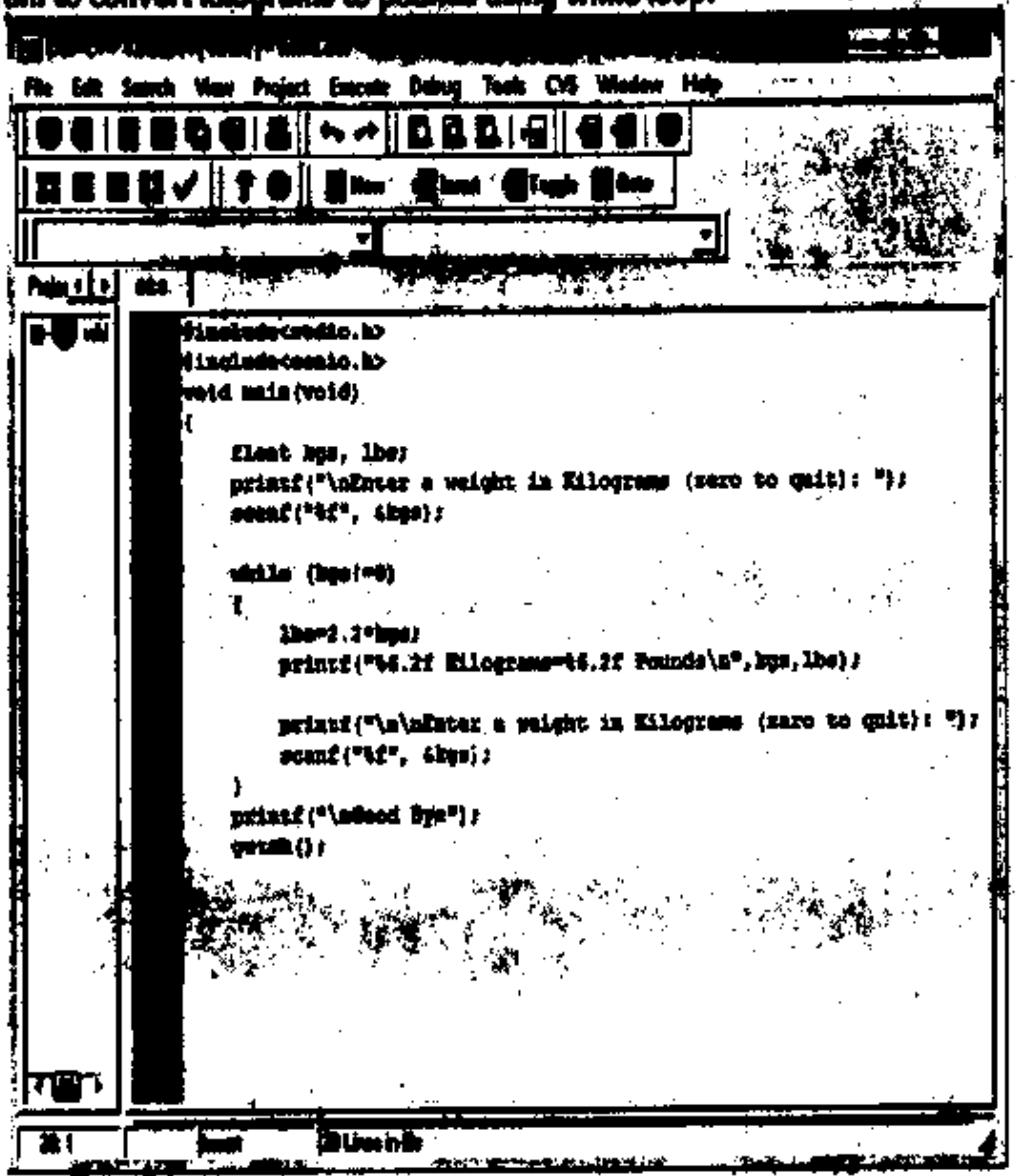


```
Enter two numbers:30,50

GCD=10
```

Execution of Program

viii. Write a program to convert kilograms to pounds using while loop.

Ans: Program to convert kilograms to pounds using while loop:

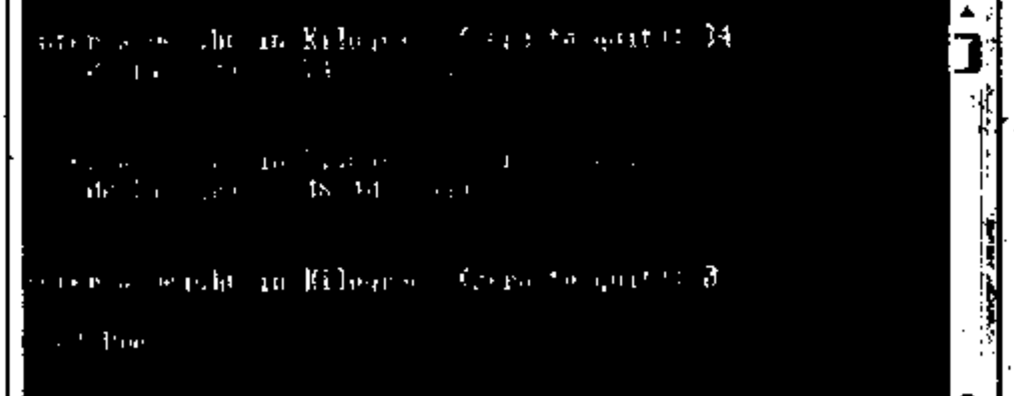


File Edit Search View Project Execute Debug Tools CVS Window Help

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    float kgs, lbs;
    printf("\nEnter a weight in Kilograms (zero to quit): ");
    scanf("%f", &kgs);

    while (kgs!=0)
    {
        lbs=2.2*kgs;
        printf("%6.2f Kilograms=%6.2f Pounds\n",kgs,lbs);

        printf("\n\nEnter a weight in Kilograms (zero to quit): ");
        scanf("%f", &kgs);
    }
    printf("\nGood Bye");
    getch();
```

Program to convert kilograms to pounds

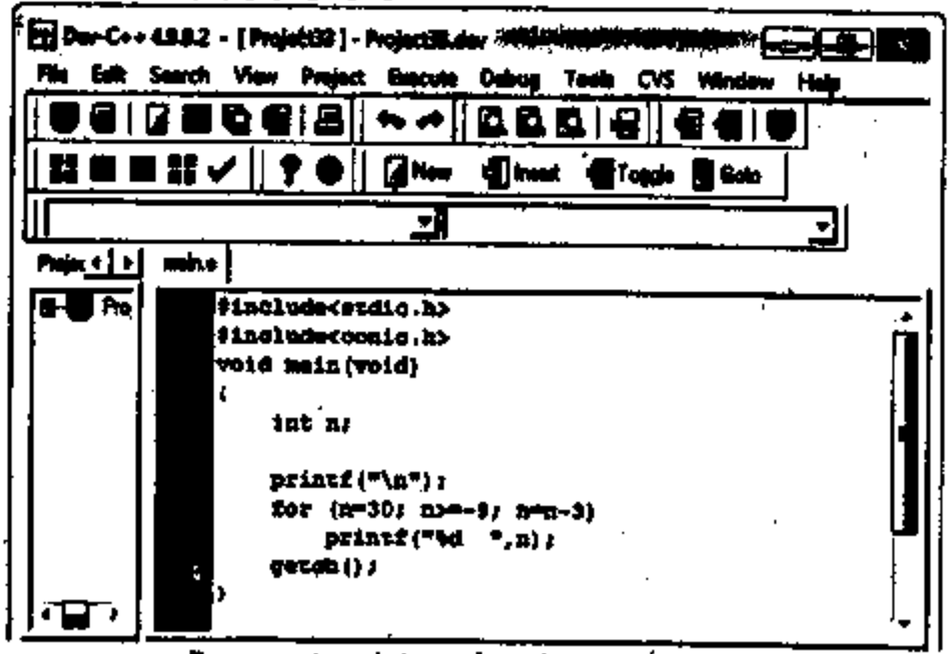
Execution of the programs is shown in Figure.



```
Enter a weight in Kilograms (zero to quit): 34
[illegible]


[illegible]
[illegible] 18.34 [illegible]


Enter a weight in Kilograms (zero to quit): 0

Good Bye
```

ix. Write a program to print the given sequence of numbers on a single line in reverse order. 30 27 24 21 18 15 12 9 6 3 0 -3 -6 -9 by using for() loop.

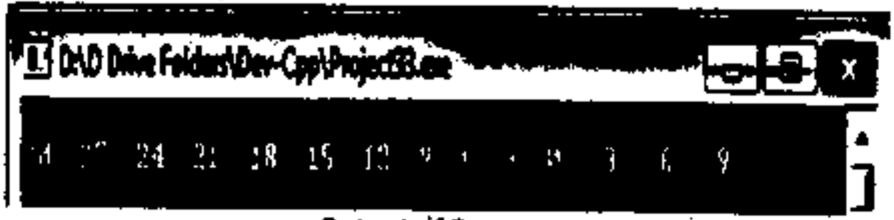
Ans: Program to print the given sequence of numbers on a single line in reverse order.
30 27 24 21 18 15 12 9 6 3 0 -3 -6 -9


Dev-C++ 4.9.9.2 - [Project33] - Project33.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int n;

    printf("\n");
    for (n=30; n>=-9; n=n-3)
        printf("%d  ",n);
    getch();
}
```

Program to print numbers in reverse order

The output of program is shown in Fig.

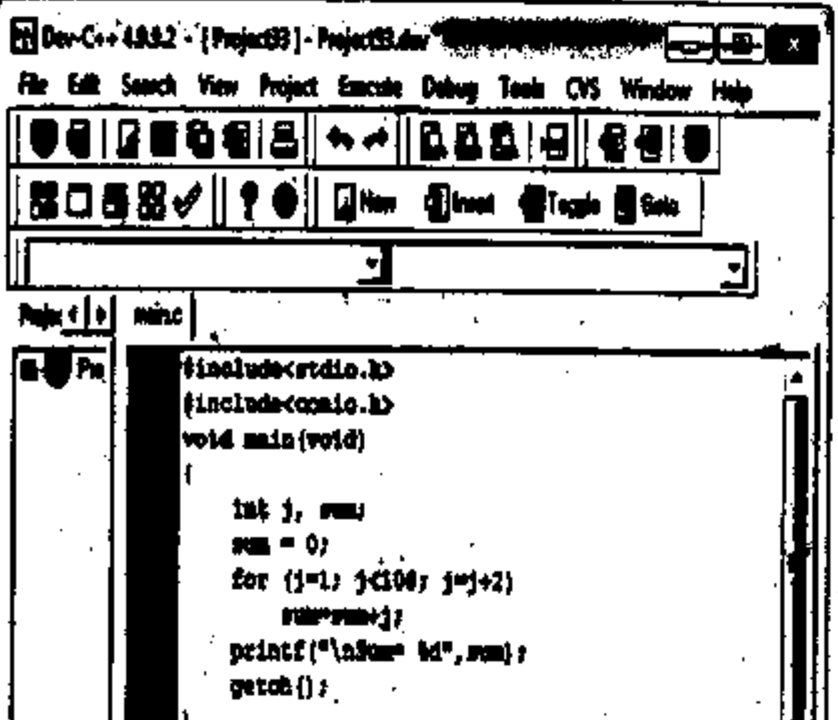


```
30  27  24  21  18  15  12  9  6  3  0  -3  -6  -9
```

Output of Program

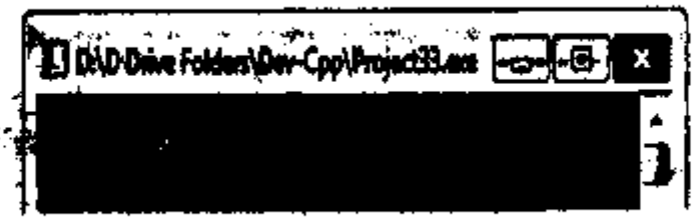
x. Write a program to print the sum of odd numbers from 1 to 100 by using for() loop.

Ans: Program to print the sum of odd numbers from 1 to 100:


Dev-C++ 4.9.9.2 - [Project33] - Project33.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int j, sum;
    sum = 0;
    for (j=1; j<100; j=j+2)
        sum=sum+j;
    printf("\nSum= %d",sum);
    getch();
}
```

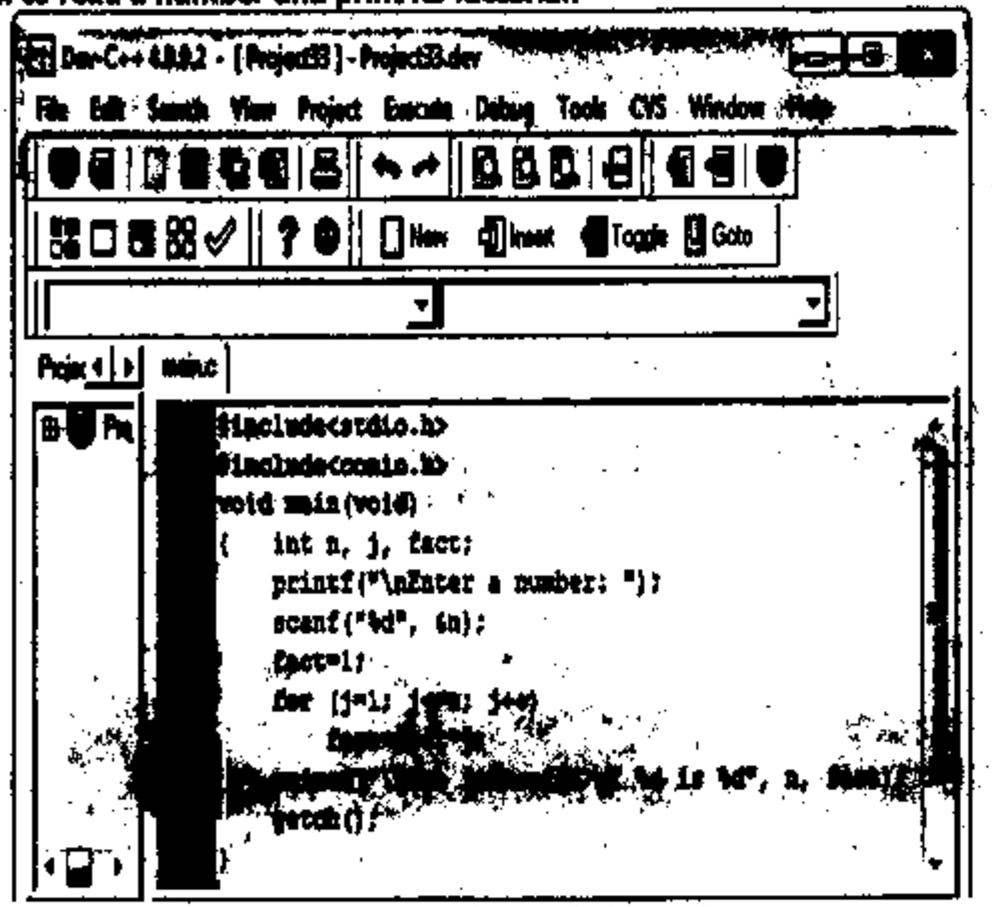
- When this program is executed, the variable sum is initialized to zero and the loop variable j to 1.
- The first value of j which is 1 is added to sum. Then j is incremented by 2 and the number 3 is added.
- On each pass through the loop the value of j is incremented by 2 and added to the previous sum. This continues till the last odd number 99 is added to the sum. The output of this program is shown is Fig. This program does not require any input from the keyboard.



Output of Program

xi. Write a program to read a number and print its factorial by using for() loop.

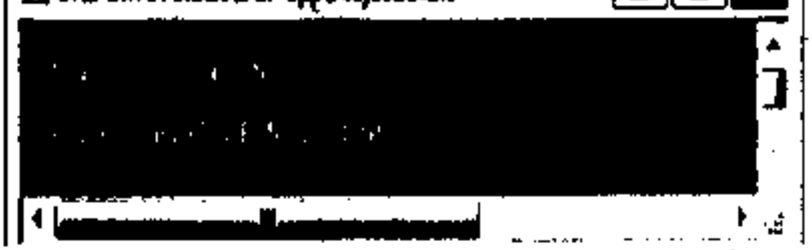
Ans: Program to read a number and print its factorial:


Dev-C++ 4.9.9.2 - [Project33] - Project33.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Projec
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{   int n, j, fact;
    printf("\nEnter a number: ");
    scanf("%d", &n);
    fact=1;
    for (j=1; j<=n; j++)
        fact[illegible]
    [illegible] %d is %d", n, fact);
    getch();
}
```

Program to print factorial of a number.

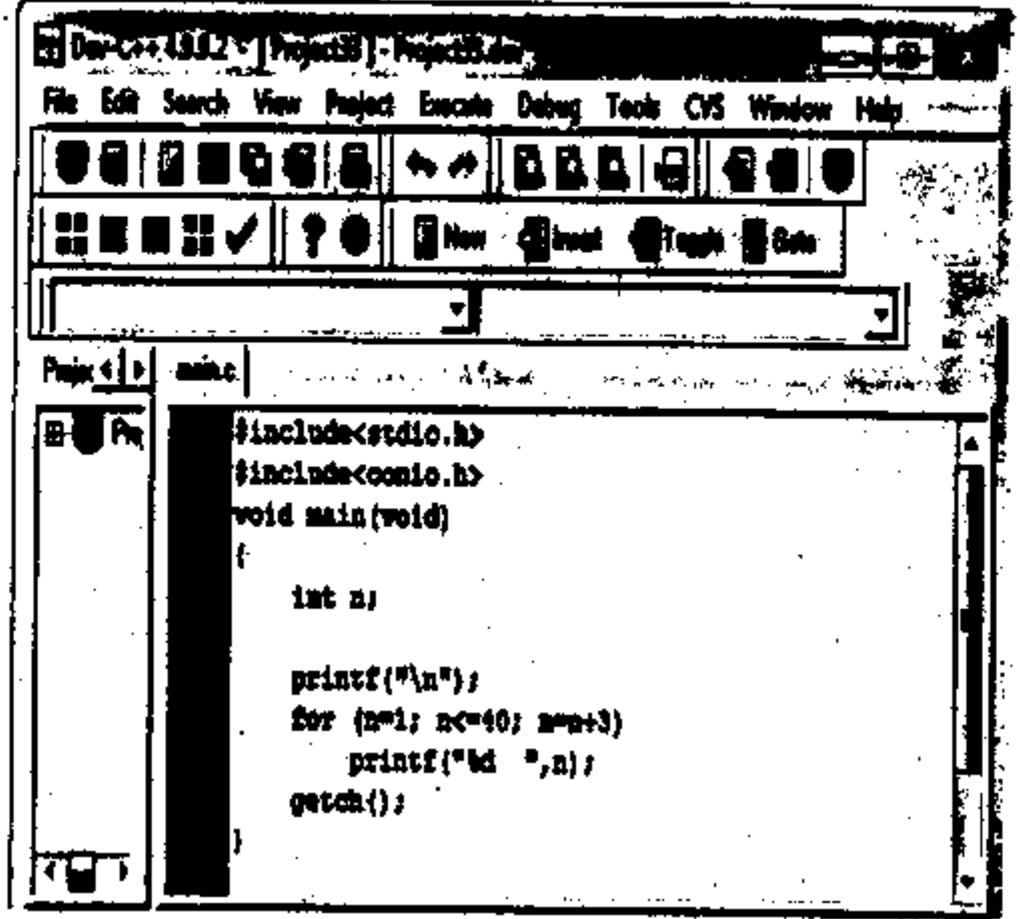
Execution of the program is shown in Fig.



Output of Program

xii. Write a program to print the given sequence of numbers on a single line.
1 4 7 10 13 16 19 22 25 28 31 34 37 40 by using for() loop.

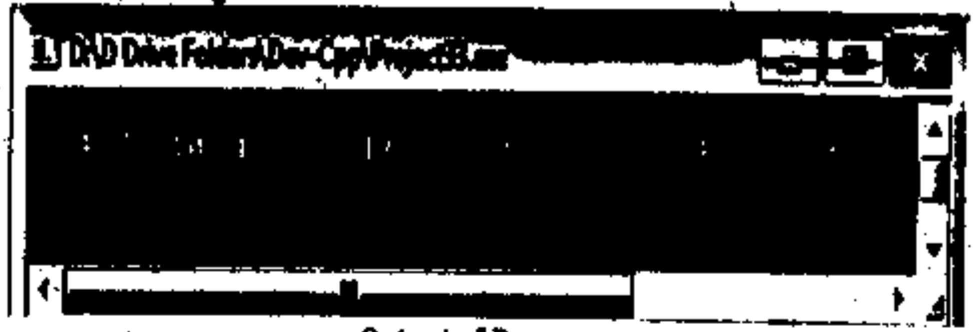
Ans: Program to print the given sequence of numbers on a single line.
1 4 7 10 13 16 19 22 25 28 31 34 37 40


Dev-C++ 4.9.9.2 - [Project38] - Project38.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Projec
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int n;

    printf("\n");
    for (n=1; n<=40; n=n+3)
        printf("%d  ",n);
    getch();
}
```

Program to print a sequence of numbers

Output of Program is shown in Fig.



Output of Program

xiii. What is loop? List essential elements of a loop. Why Programmer might want to execute a loop.

Ans: Loop:

A loop is a statement in a programming language that allows one or more statements to be repeatedly executed as many times as required.

Essential elements of a loop:

There are two essential elements of a loop. The block of statements forms the body of the loop that is to be executed repeatedly until loop condition is true. Loops terminate based on test conditions.

Programmer might want to execute a loop:

I. a given number of times
II. until a given value exceeds another value

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. **(2 × 8 = 16)**

Q3. What is looping structure? Explain for loop with examples.

Ans: Loop Structure:

A loop is a structure that enables the programmer to execute the same sequence of statements repeatedly until a particular condition is met.

For loop / For Statement:

The for is a looping statement which is used to execute a set of statements repeatedly for a fixed number of times. It is also known as counter loop. It has the general form:

```
for (initialization; test condition; increment/decrement).
{
Body of the loop
}
```

When for statement is executed, a variable (also known as loop variable) is assigned an initial value in the initialization part of the loop, such as k=1 or count=0. The value of the loop variable is checked with the given test condition. The test condition is a relational expression, such as k<10. If the condition is true, the control enters the body of the loop otherwise it will exit the loop,

After the execution of the body of the loop, the control is transferred back to the increment/decrement part of the loop. The loop variable is incremented or decremented using an assignment statement such as $k = k + 1$. The new value of loop variable is again checked with the test condition. If the condition is satisfied then the body of the loop is again executed. This process goes on till the test condition becomes false.

Body of the loop may have one or more statements. If it contains only a single statement then braces are not needed.

Note: All the three expressions such as initialization, test condition and increment are optional. You can omit any or all in for statement.

Example: For example the following all for statements are valid.

```
for(; ;)
for(int i=1 ; ;)
for(; k<10 ; k++)
for(; ; k++)
for(; x<12 ;)
```

Q4. Explain while and do-while loops with examples.

Ans: While loops/ The While Statement:

A repetition structure when the number of iterations is not known in advance and the repetition continues until test condition remains true.

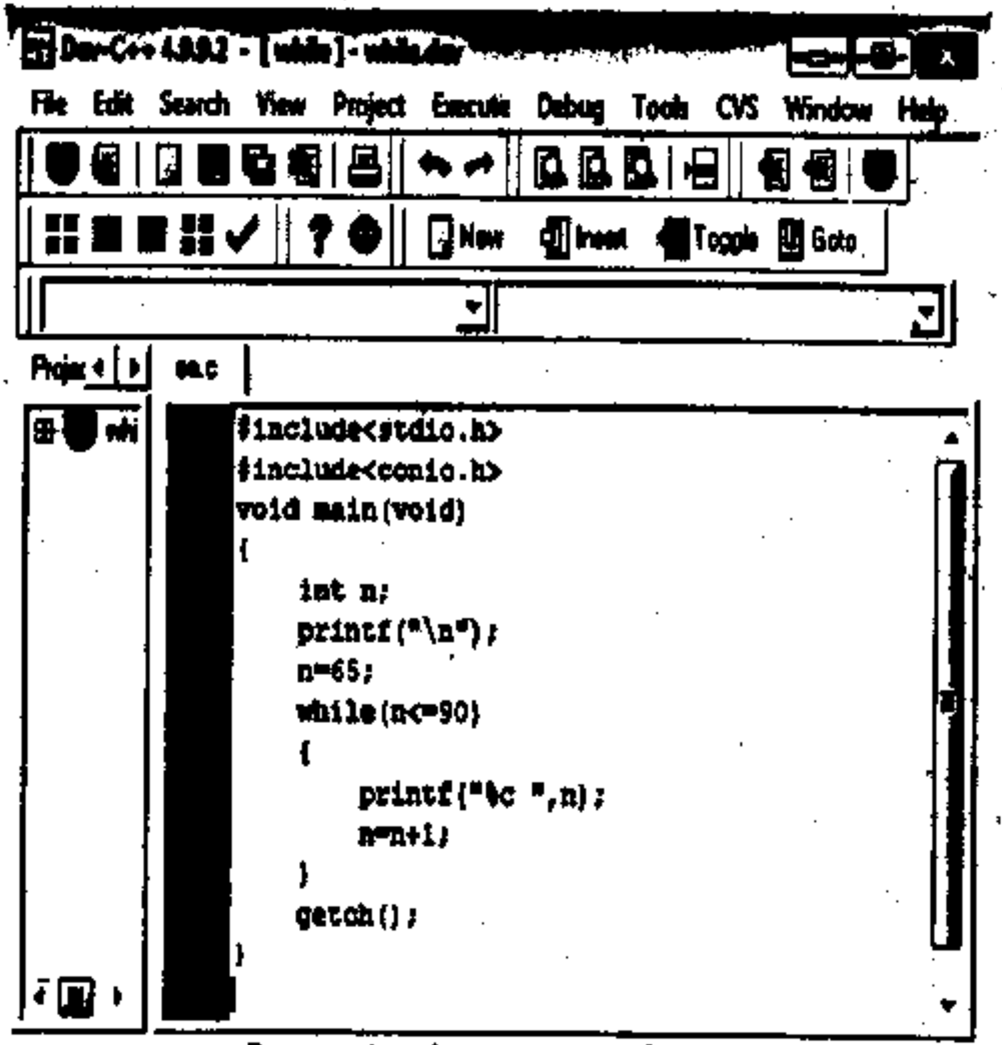
The while statement has the general form:

```
while (test condition)
{
        Body of the loop
}
```

When a while statement is executed, the computer first evaluates the test condition. If it is true, body of the while loop is executed. After the execution of the body of the loop, the test condition is again evaluated and if it is true, the body of the loop is executed once again. This process continues until the test condition becomes false. When it becomes false, the control is transferred to the first statement following the end of body of loop.

The body of the loop can be a single statement or it can be multiple statements. If the body of the loop consists of a single statement then the braces are not required but if it consists of more then one statement then braces must be used.

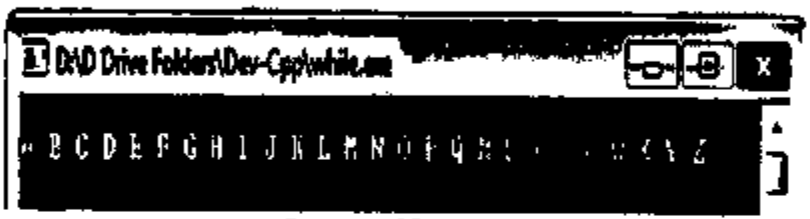
Examples: Program:


Dev-C++ 4.9.9.2 - [while] - while.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Projec
ee.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int n;
    printf("\n");
    n=65;
    while(n<=90)
    {
        printf("%c ",n);
        n=n+1;
    }
    getch();
}
```

Program to print upper – case letters

Output of the program is shown in Figure.



```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Output of Program

Do-while loops / The Do While Statement:

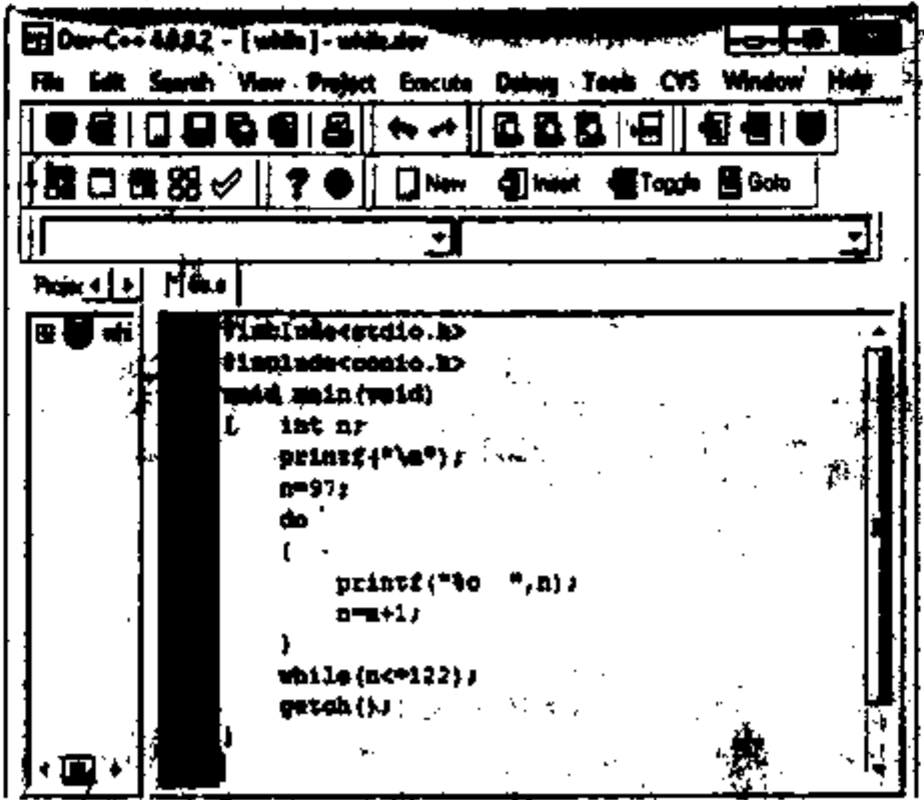
The do while statement is used to implement loop structure when it is required to execute the loop at least once. The general form of the **do while** loop is given below.

```
do
{
        Body of the loop
}
while (test condition);
```

The statement **while (test condition)** is placed at the end of the loop so that the body of the loop is executed at least once whether the condition is true or false. There is a semicolon after the test condition because it is at the end of loop. If the body of loop contains a single statement then braces are not required.

Program:

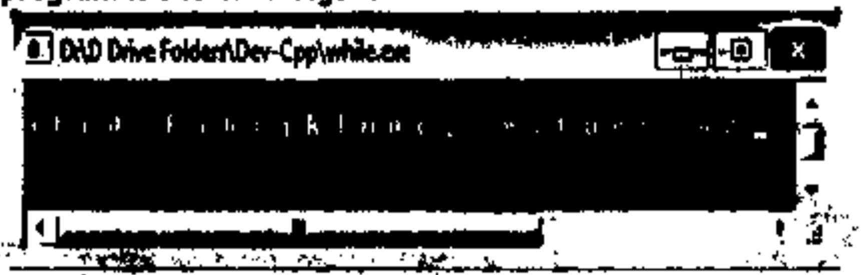
The program in Figure, prints all the lower-case letters of the alphabet using do while loop.


Dev-C++ 4.9.9.2 - [while] - while.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project

```
#include<stdio.h>
#include<conio.h>
void main(void)
{   int n;
    printf("\n");
    n=97;
    do
    {
        printf("%c  ",n);
        n=n+1;
    }
    while(n<=122);
    getch();
}
```

Program to print lower – case letters

The output of this program is shown in Figure.



```
a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
```

Output of Program

- In this program, the integer variable n is initialized to 97 which is the ASCII value of letter a, before entering the loop.
- The expression in the do-while statement, n<=122 acts as the test condition. The statements within the loop execute as long as the expression remains true.
- The statement n=n+1 increments the value of n by one with each iteration. Note that the ASCII value of z is 122.

Q5. What is nested [illegible]

Ans: Nested Loops:

In C language, it is allowed to nest loops within another loop. Nested loops can be of any kind. For example, the programmer can nest a for loop inside a while loop or inside a do while loop. Loops can be mixed in any way as required.

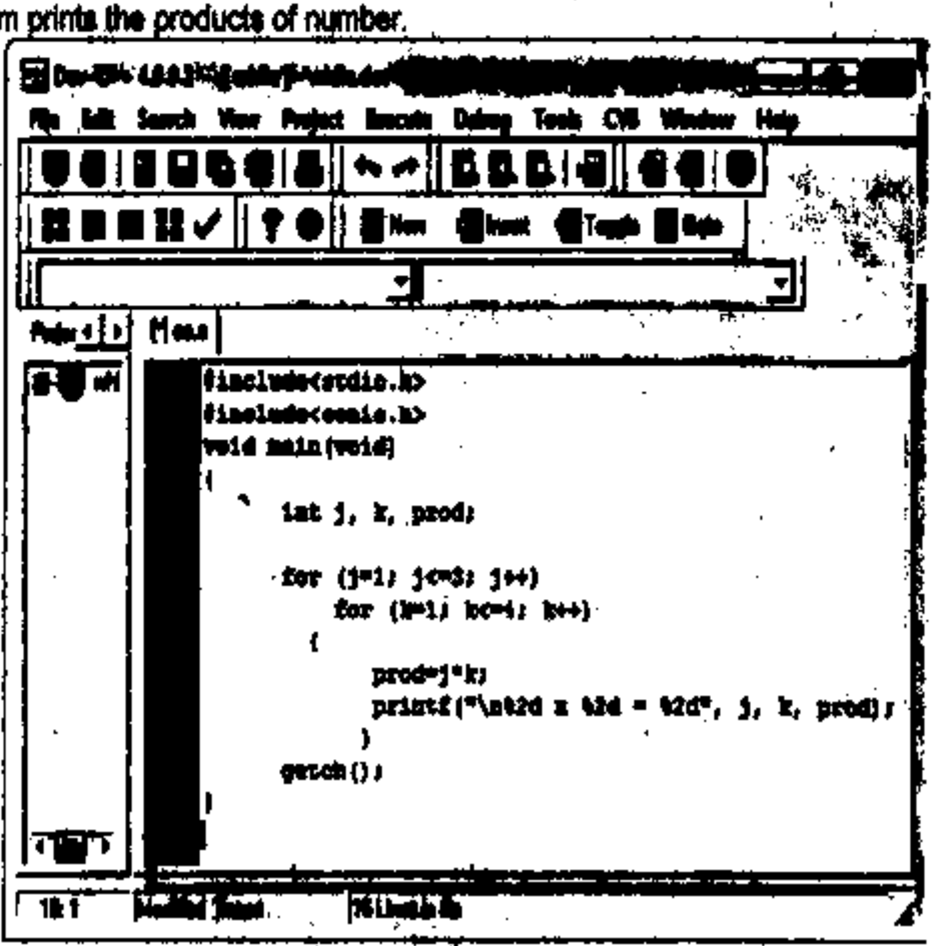
Example # 1:

Program uses nested loop to print the products of numbers as given below.

1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
1 × 4 = 4
2 ×1 = 2
2 × 2 = 4
2 × 3 = 6

3 ×3 = 9
3 × 4 = 12

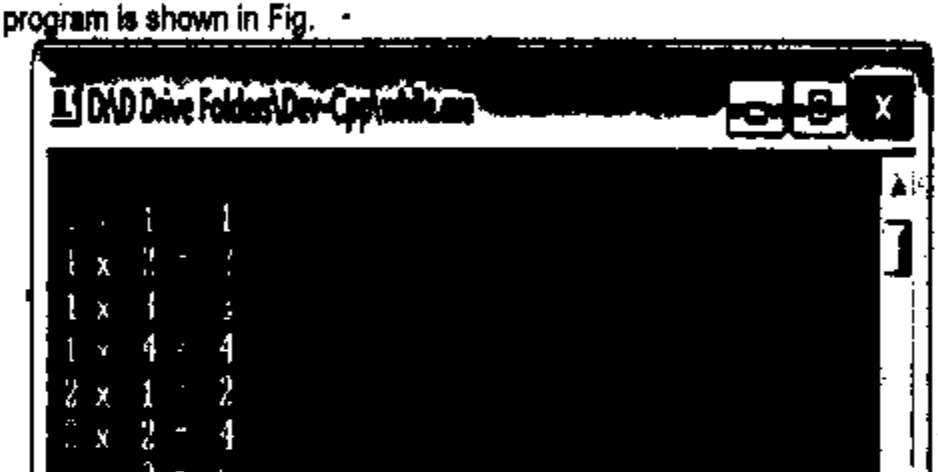
The program prints the products of number.



```
#include<stdio.h>
#include<conio.h>
void main(void)
{
        int j, k, prod;

        for (j=1; j<=3; j++)
            for (k=1; k<=4; k++)
          {
              prod=j*k;
              printf("\n%2d x %2d = %2d", j, k, prod);
            }
        getch();
}
```

Program to print products using nested loop

The output of the program is shown in Fig.



```
 1 x  1 =  1
 1 x  2 =  2
 1 x  3 =  3
 1 x  4 =  4
 2 x  1 =  2
 2 x  2 =  4
 2 x  3 =  6
 2 x  4 =  8
 3 x  1 =  3
 3 x  2 =  6
 3 x  3 =  9
 3 x  4 = 12
```

Output of Program

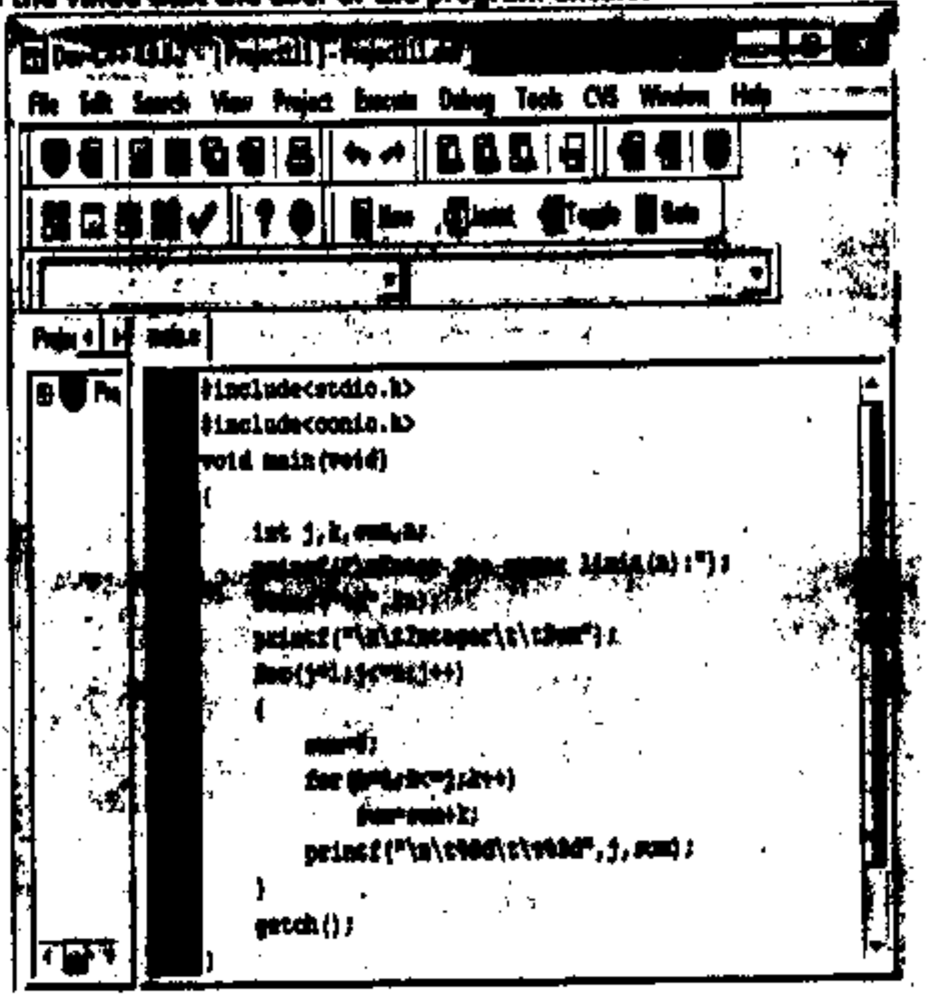
```
[illegible] x %3d = %3d", j, k, prod);
}
```

is executed. This calculates and displays the first four products, 1×1, 1×2, 1×3 and 1×4.

- The value of j is then incremented by 1 and the inner loop is executed again. This calculates and displays the next four products, 2×1, 2×2, 2×3 and 2×4.
- Finally, j is incremented to 3, giving the last four products, 3×1, 3×2, 3×3 and3 × 4.
- In this program braces must be used in the inner for loop because there are more than one statements to be executed.

Example # 2:

Program uses nested loop to calculate the sum of the integers for each integer from 1 to n, where n is the value that the user of the program enters:

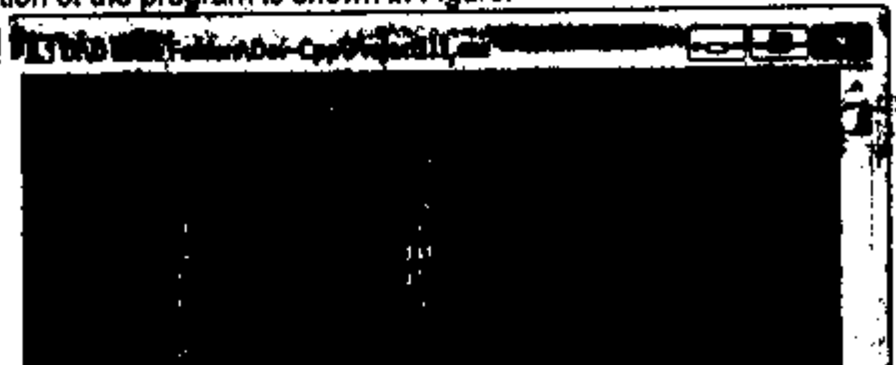


File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int j,k,sum,n;
    printf("\nEnter the upper limit(n):");
    scanf("%d",&n);
    printf("\n\tInteger\t\tSum");
    for(j=1;j<=n;j++)
    {
        sum=0;
        for (k=1;k<=j;k++)
            sum=sum+k;
        printf("\n\t%6d\t\t%6d",j,sum);
    }
    getch();
}
```

Program to print sum of numbers

The execution of the program is shown in Figure.



IMPORTANT QUESTIONS & ANSWERS

Q1. Highlight the function of loops.

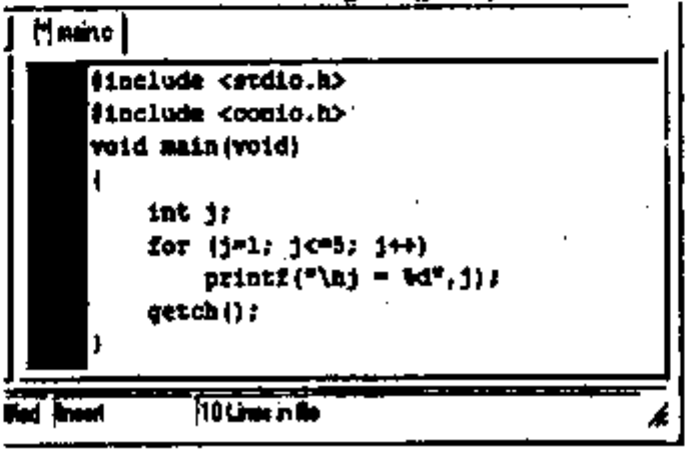
Ans: Function of Loop:

Loops are used in programs to repeat a block of statements. Repeating a block of statements is a very common and useful task in programming. Loops make the task of writing programs easier and efficient.

Three types of loops are used in C programs. These are **for, while,** and **do while** loops.

Q2. Write a program to print numbers from 1 to 5 by using for() loop.

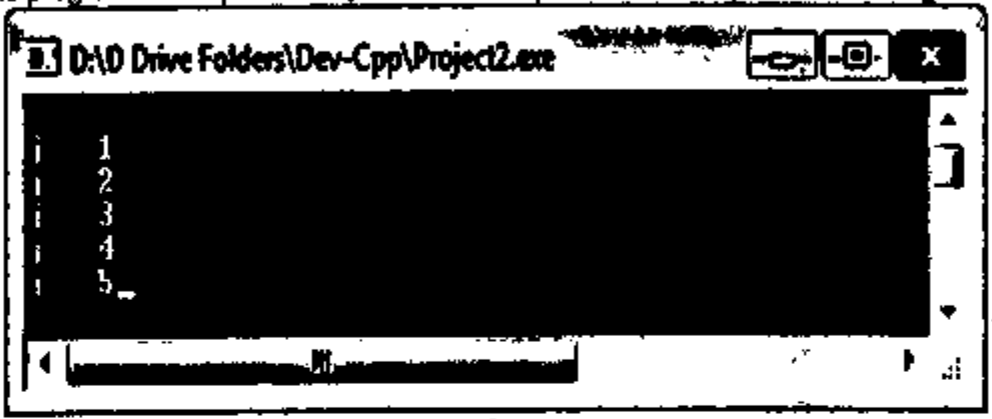
Ans: Program to print numbers from 1 to 5 using for() loop.



```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int j;
    for (j=1; j<=5; j++)
        printf("\nj = %d",j);
    getch();
}
```

Program to print numbers from 1 to 5

When this program is compiled and executed it will produce the output shown in Fig.



D:\D Drive Folders\Dev-Cpp\Project2.exe

```
j    1
j    2
j    3
j    4
j    5_
```

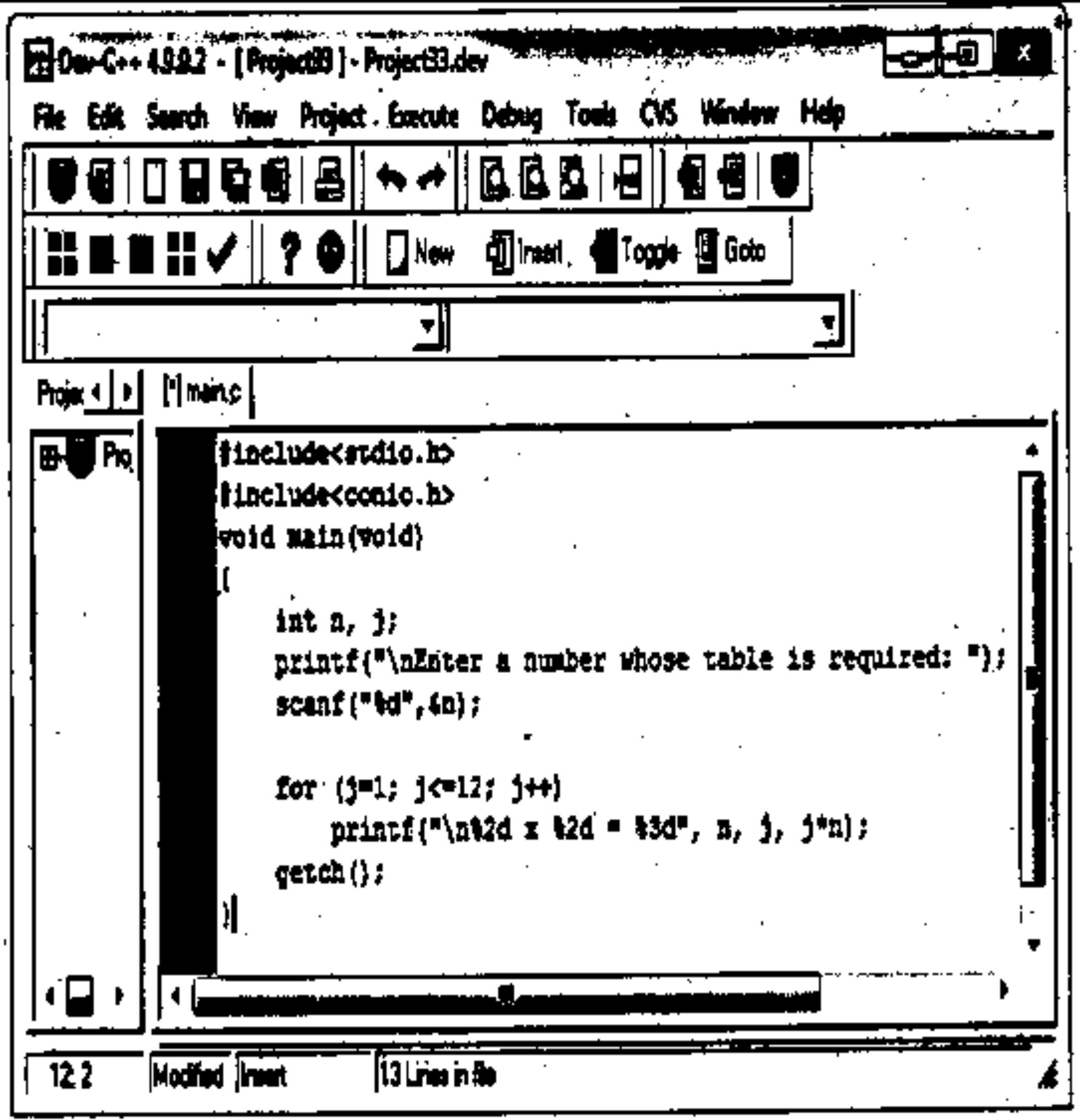
Output of program

- The expression **j=1** inside the parenthesis initializes the j variable to 1. Initialization is done as soon as the loop is entered.
- The second expression (test condition) **j<= 5,** tests each time through the loop to see if j is less than or equal to 5. If the test is true, the body of the loop is executed. If the test is false, the loop will be terminated and control will be transferred to the next statement following the for loop. In this program there are no more statements, so the entire program terminates.
- The third expression (increment/decrement) j++ increments the loop variable j, each time the loop is executed. In general, any expression can be used for incrementing the loop variable. When a for loop terminates, the loop variable is still defined and contains the value assigned by the last increment. In Program 1, the last value assigned to j will be 6. However, a well-designed program will not use this feature.

Q3. Write a program to read a number and print its table by using for() loop.

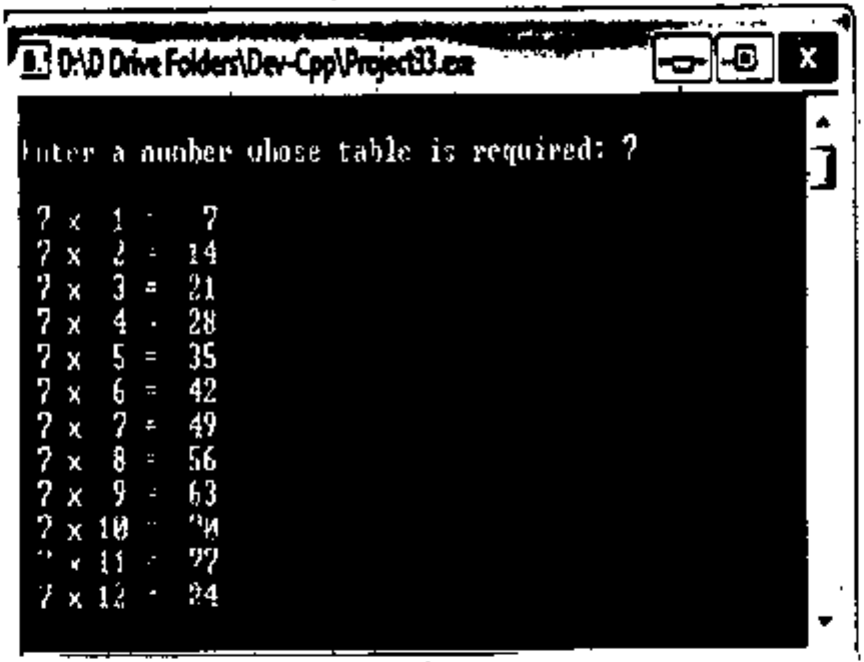
Ans: Program to read a number and print its table using for() loop:

- The execution of program is shown in Figure, when the value of n is 7. The program will print the value


Dev-C++ 4.9.9.2 - [Project33] - Project33.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
[*] main.c
12: 2 Modified Insert 13 Lines in file

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int n, j;
    printf("\nEnter a number whose table is required: ");
    scanf("%d",&n);

    for (j=1; j<=12; j++)
        printf("\n%2d x %2d = %3d", n, j, j*n);
    getch();
}
```

Program to print table of a number



```
Enter a number whose table is required: 7

 7 x  1 =   7
 7 x  2 =  14
 7 x  3 =  21
 7 x  4 =  28
 7 x  5 =  35
 7 x  6 =  42
 7 x  7 =  49
 7 x  8 =  56
 7 x  9 =  63
 7 x 10 =  70
 7 x 11 =  77
 7 x 12 =  84
```

Output of Program

- In the printf() function, %2d means to print the values of n and j in field width of 2 and %3d means to

GUESS PAPER & MODEL PAPER # 6

Based on Chapter # 06 (Reduced Syllabus)

COMPUTER LOGIC AND GATES

SECTION-A (Marks 12)

Time allowed: 20 Minutes Marks: 12

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. Which operation is represented by "+" sign?
A. AND B. OR C. NOT D. NAND

ii. Which operation is represented by a dot or absence of an operator?
A. AND B. OR C. NOT D. NAND

iii. Which of the following gates is also known as inverter?
A. OR gate B. NOR gate
C. NAND gate D. NOT gate

iv. Which combination of inputs to a two input AND gate will produce output of HIGH?
A. LOW and LOW B. LOW and HIGH
C. HIGH and HIGH D. None of these

v. Which logic gate is represented by the function, $F = \overline{(x + y)}$?
A. NAND B. NOR
C. Exclusive-OR D. Exclusive-NOR

vi. Which logic gate is represented by the function, $= \overline{(xy)}$?
A. NAND B. NOR
C. Exclusive-OR D. Exclusive-NOR

vii. What is the maximum number of possible input combinations in a truth table that has three variables?
A. 3 B. 6 C. 8 D. 9

viii. How many AND gates are required to create the logic circuit of the Boolean function: $F = \bar{x}z + y\bar{z} + xyz$?
A. 1 B. 2 C. 3 D. 4

ix. Which of the following gate gives the output as 1 only if all the inputs signals are 1?
A. AND B. OR C. EXOR D. NOR

x. The boolean expression of an OR gate is:
A. A.B B. A'B+AB' G. A+B D. A'B'

xi. The universal gate that can be used to implement any Boolean expression is:
A. NAND B. EXOR C. OR D. AND

xii. The gate which is called an inverter is called __________.
A. NOR B. NAND C. EXOR D. NOT

COMPUTER SCIENCE SSC-II

Time allowed: 2:40 Hours Total Marks Sections B and C: 43

SECTION – B (Marks 27)

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. What is a logic gate?

ii. Define truth table.

iii. Define boolean function.

iv. Draw the logic circuits of the following Boolean functions. $F_1 = \bar{x}\,\bar{y}z + x\bar{y}z$

v. Draw the logic circuits of the following Boolean functions. $F_2 = \bar{x} + yz$

vi. Draw the logic circuits of the following Boolean functions. $F_3 = x\bar{y} + x\bar{y}z + xyz$

vii. Draw the logic circuits of the following Boolean functions. $F_4 = x + \bar{y} + y\bar{z}$

viii. Describe the properties of truth table.

ix. Describe AND gate. Draw the truth table for AND gate.

x. Describe OR gate. Draw the truth table for OR gate.

xi. Describe NOT gate. Also, draw truth table for NOT gate.

xii. What is boolean expression and boolean function? Give its examples.

xiii. Explain the conversion of boolean function $F_2 = x + \bar{y}z$ **to logic circuit.**

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. Draw the graphical symbols of AND, OR, NOT, NAND and NOR gates and write their functions.

Q4. Explain how NAND and NOR gates can be created using AND, OR and NOT gates.

Q5. Draw truth table of the following Boolean functions.

i) $F_1 = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z}$ ii) $F_2 = \bar{x}z + y\bar{z} + xyz$

iii) $F_3 = \overline{xy}\bar{z} + \bar{x}\bar{y}z + \bar{x}yz + xy\bar{z}$ iv) $F_4 = x\bar{z} + \overline{xy}$

SOLUTION OF GUESS PAPER & MODEL PAPER # 6 (Reduced Syllabus)

SECTION- A (MCQs)

i. B	ii. A	iii. D	iv. C	v. B	vi. A
vii. C	viii. C	ix. A	x. C	xi. A	xii. D

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. What is a logic gate?

Ans: Logic Gates:

Logic gates are the basic building blocks of digital computer. Logic gates operate on two voltage levels and process digital signals which represent binary digits 0 and 1.

ii. Define truth table.

Ans: Truth Table:

A truth table represents a digital logic circuit in table form. It shows how a logic circuit's output responds to all the possible combinations of the inputs using logic '1' for true and logic '0' for false.

iii. Define boolean function.

Ans: Boolean function:

A Boolean function can be transformed from an algebraic expression into a logic circuit composed of AND, OR and NOT gates.

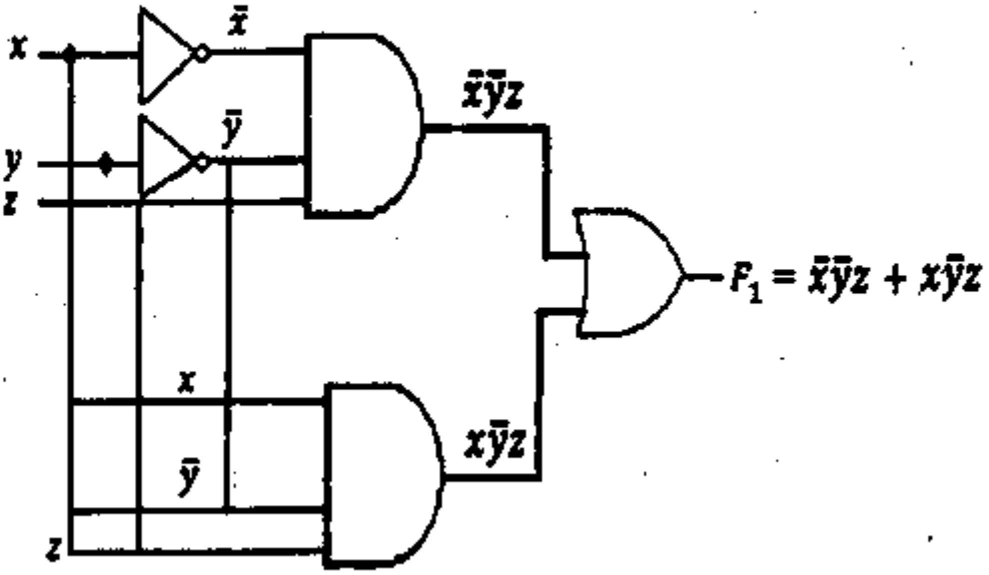
Example: As an example, consider the following Boolean function.

$F = x + y$

The function F is equal to 0, if $x = 0$ and $y = 0$. For all the other combinations of x and y, the function will be equal to 1.

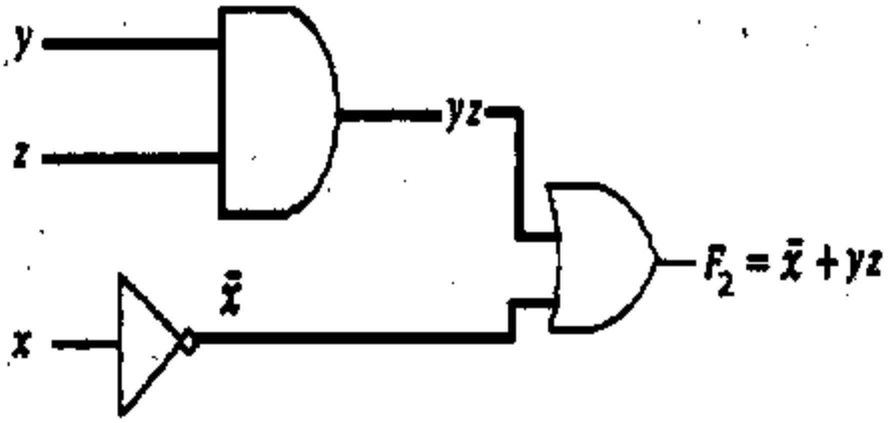
iv. Draw the logic circuits of the following Boolean functions. $F_1 = \bar{x}\,\bar{y}z + x\bar{y}z$

Ans:



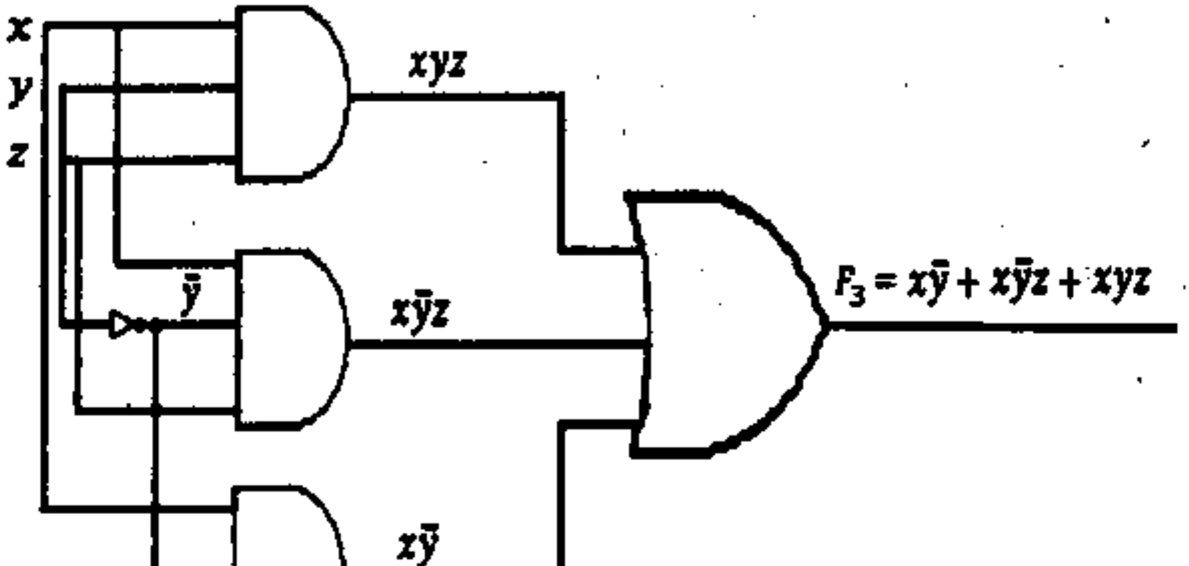
v. Draw the logic circuits of the following Boolean functions. $F_2 = \bar{x} + yz$

Ans:



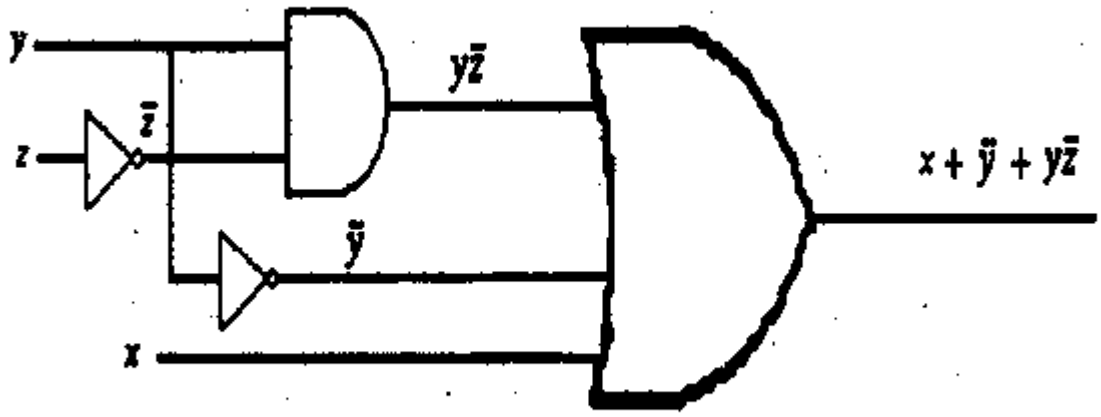
vi. Draw the logic circuits of the following Boolean functions. $F_3 = x\bar{y} + x\bar{y}z + xyz$

Ans:



vii. Draw the logic circuits of the following Boolean functions. $F_4 = x + \bar{y} + y\bar{z}$

Ans:



viii. Describe the properties of truth table.

Ans: Properties of Truth Table:

It has the following properties.

- Truth table consists of rows and columns.
- It shows relationship between inputs to and output from a digital logic circuit.
- It shows output for all the possible combinations of inputs using 0 for LOW and 1 for HIGH.
- All the combinations of inputs are listed in columns on the left and output is shown in the right most column.
- The input columns are constructed in the order of binary counting with a number of bits equal to the number of inputs.
- Each row of truth table contains one possible combination of inputs and the result of the operation for those values is shown in the right most column.

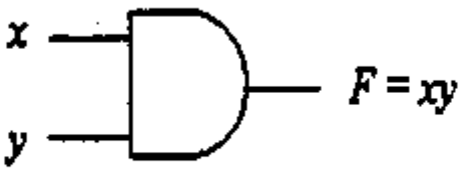
ix. Describe AND gate. Draw the truth table for AND gate.

Ans: AND Gate:

AND gate is a physical realization of the logical multiplication. It is the implement of AND operation.

Explanation:

The AND gate operates such that the output will be at level 1 (HIGH) only when all inputs are 1 (HIGH). The mathematical expression for the two-input AND gate is written as F=xy. For a three-input gate, it would be F=xyz, and so on for more inputs.



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

Symbol, expression and truth table of AND gate

Truth table for AND gate

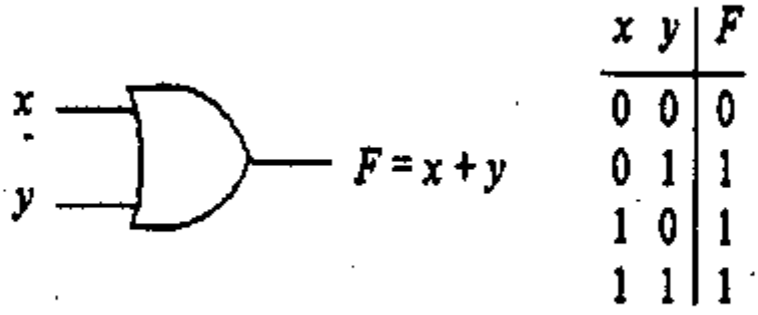
x	y	$Output = F = xy$
0	0	0 . 0 = 0
0	1	0 . 1 = 0
1	0	1 . 0 = 0
1	1	1 . 1 = 1

Describe OR gate. Draw the truth table for OR gate.

Explanation:

The OR gate produces a 1 output when any input is 1.

Its mathematical expression is F = x+y, where the + stands for the OR operation and not normal addition. For a three-input OR gate, it would be: F = x+y+z, and so on.



Symbol, expression and truth table of OR gate

Truth table for OR gate

x	y	$Output = F = x + y$
0	0	0 + 0 = 0
0	1	0 + 1 = 1
1	0	1 + 0 = 1
1	1	1 + 1 = 1

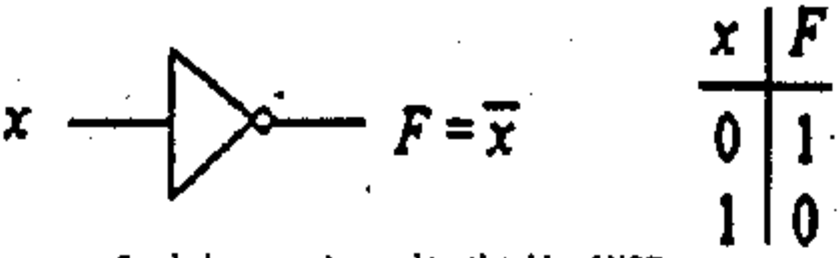
xi. Describe NOT gate. Also, draw truth table for NOT gate.

Ans: NOT gate:

NOT gate is a physical realization of the complementation operation.

Explanation:

A NOT gate is a single input gate. It converts LOW to HIGH and vice versa. Therefore, it is commonly known as an *inverter*. Its logic expression is $F = \overline{x}$. The bar in the expression indicates the inversion operation. In the output of graphical symbol, the small circle indicates inversion.



Symbol, expression and truth table of NOT gate

Truth table for NOT gate

x	Output = F = $\bar{x}$
0	0 = 1
1	1 = 0

xii. What is boolean expression and boolean function? Give its examples.

Ans: A Boolean Expression and Boolean Function:

A Boolean expression is composed of a combination of the Boolean constants (True or False), Boolean variables and logical operators (AND, OR, NOT).

A Boolean expression always produces a Boolean value (1 or 0).

Each Boolean expression represents a Boolean function.

Examples of Boolean expression/function:

Some examples of Boolean expression/function are:

$F = x + y + \bar{z}$

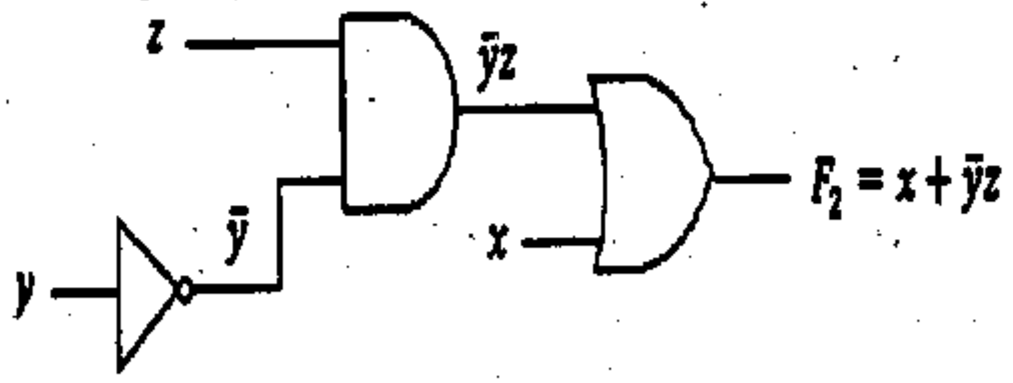
$F = (x.y) + \bar{z}$

$F = (x + y) + (x + \overline{y})$

xiii. Explain the conversion of boolean function $F_2 = x + \bar{y}z$ to logic circuit.

Ans: Conversion of Boolean Function $F_2 = x + \bar{y}z$ to Logic Circuit:

To create the logic circuit of this function, one AND gates is required for the term $\bar{y}z$, one NOT gate to convert y to $\bar{y}$ and an OR gate to perform OR operation on the terms x and $\bar{y}z$.



Logic circuit of Function F_2

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. Draw the graphical symbols of AND, OR, NOT, NAND and NOR gates and write their functions.

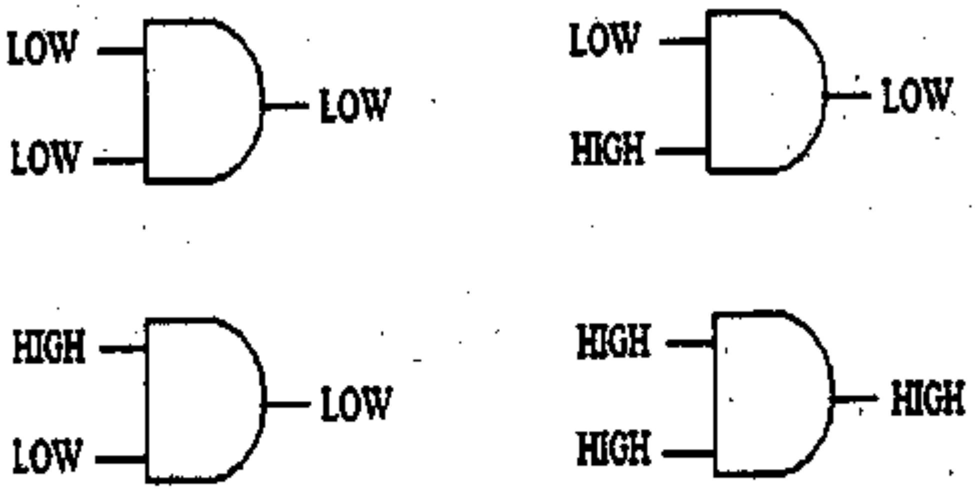
Ans: Basic Logic Gates:

There are three basic logic gates used in digital circuits which are AND, OR and NOT gates.

- **AND Gate:**

The AND gate has two or more inputs that can be LOW (0) or HIGH (1). The output is HIGH only when all the inputs are HIGH. It produces a LOW output when at least one of the inputs is LOW.

The logic operation of a two-input AND gate is shown in Fig., with all the possible input combinations and the resulting output for each.

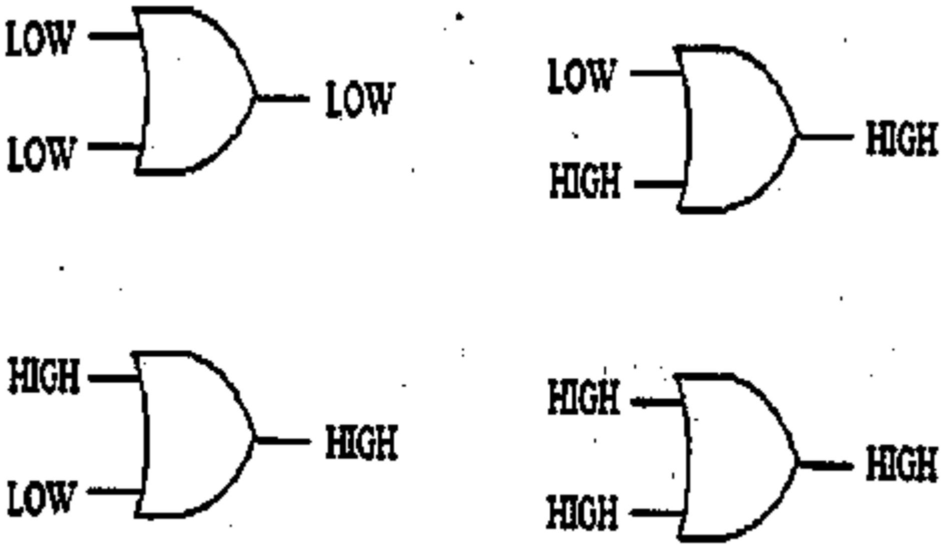


The AND gate

- **OR Gate:**

The OR gate has two or more inputs. The output of an OR gate is LOW only when all the inputs are LOW. The output is HIGH when one or more of its inputs are HIGH.

The logic operation of a two-input OR gate is shown in Fig., with all the possible input combinations and the resulting output for each.

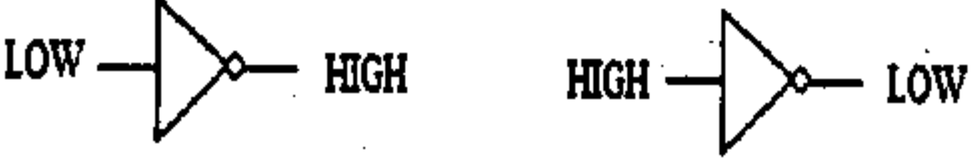


The OR gate

- **NOT Gate:**

The NOT gate performs the functions of inversion. Therefore, it is also known as inverter. It has a single input.

The output of a NOT gate is HIGH when the input is LOW and vice versa.



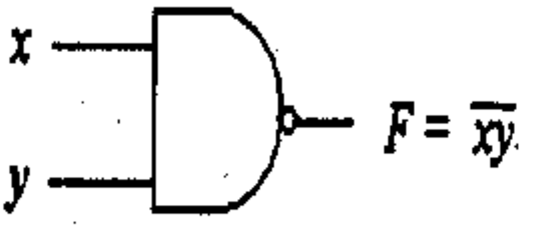
The NOT gate

- **The NAND Gate:**

The NAND gate combines the AND and NOT gates, such that the output will be 0 only when all the inputs are 1 as shown in Fig.

Its logic expression is $F = \overline{xy}$ which indicates that inputs x and y are first ANDed and then the result is inverted. Inversion is indicated by a bar.

Thus, an AND gate always produces an output that is the inverse (opposite) of an AND gate.

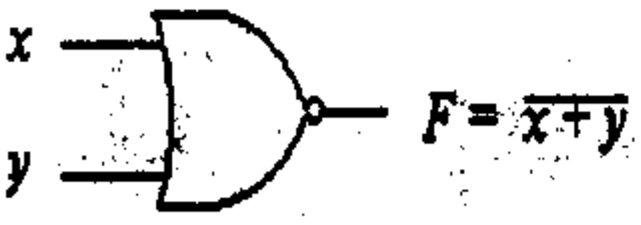


x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

Symbol, expression and truth table of NAND gate

- **The NOR Gate:**

The NOR gate combines the OR and NOT gates, such that the output will be 0 when any input is 1 as shown in Fig. Its logic expression is $F = \overline{x+y}$, which indicates that x and y are first ORed and then the result inverted. Inversion is Indicated by a bar.



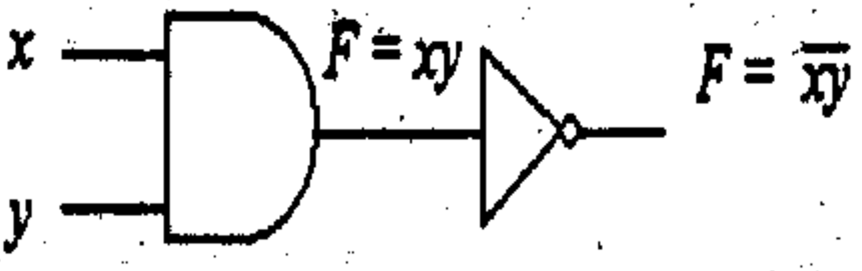
x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

Symbol, expression and truth table of NOR gate

Q4. Explain how NAND and NOR gates can be created using AND, OR and NOT gates.

Ans: Creating NAND Gate Using Basic Gates:

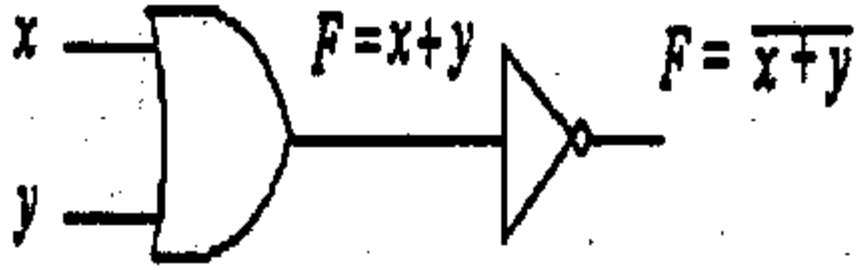
I. The NAND gate can be easily created by using an AND gate and a NOT gate as shown in Fig.



Creating a NAND gate using basic gates

Creating NOR Gate Using Basic Gates:

II. The NOR gate can also be created in a similar way by using an OR gate and a NOT gate as shown in Fig.



Creating a NOR gate using basic gates

Q5. Draw truth table of the following Boolean functions.

i) $F_1 = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z}$

ii) $F_2 = \bar{x}z + y\bar{z} + xyz$

iii) $F_3 = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}yz + xy\bar{z}$

iv) $F_4 = x\bar{z} + \bar{x}\bar{y}$

Ans: i) $F_1 = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z}$

x	y	z	$\bar{x}$	$\bar{y}$	$\bar{z}$	$\bar{x}y\bar{z}$	$\bar{x}yz$	$xy\bar{z}$	F_1
1	1	1	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	1	1
1	0	1	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
0	1	1	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0	0	1
0	0	1	1	1	0	0	0	0	0

ii) $F_2 = \overline{x}z + y\overline{z} + xyz$

x	y	z	$\overline{x}$	$\overline{y}$	$\overline{z}$	$\overline{x}z$	$y\overline{z}$	xyz	F_2
1	1	1	0	0	0	0	0	1	1
1	1	0	0	0	1	0	1	0	1
1	0	1	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
0	1	1	1	0	0	1	0	0	1
0	1	0	1	0	1	0	1	0	1
0	0	1	1	1	0	1	0	0	1
0	0	0	1	1	1	0	0	0	0

iii) $F_3 = \overline{x}\overline{y}\overline{z} + \overline{x}\overline{y}z + \overline{x}yz + xy\overline{z}$

x	y	z	$\overline{x}$	$\overline{y}$	$\overline{z}$	$\overline{x}\overline{y}\overline{z}$	$\overline{x}\overline{y}z$	$\overline{x}yz$	$xy\overline{z}$	F_3
1	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	1
1	0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0	1	0	1
0	1	0	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	1	0	0	1
0	0	0	1	1	1	1	0	0	0	1

iv) $F_4 = x\overline{z} + \overline{x}\overline{y}$

x	y	z	$\overline{x}$	$\overline{y}$	$\overline{z}$	$x\overline{z}$	$\overline{x}\overline{y}$	F_4
1	1	1	0	0	0	0	0	0
1	1	0	0	0	1	1	0	1
1	0	1	0	1	0	0	0	0
1	0	0	0	1	1	1	0	1
0	1	1	1	0	0	0	0	0
0	1	0	1	0	1	0	0	0
0	0	1	1	1	0	0	1	1
0	0	0	1	1	1	0	1	1

IMPORTANT QUESTIONS & ANSWERS

Q1. Define digital logic and logic gates.

Ans: Digital Logic:

Digital logic is fundamental in creating electronic devices such as calculator, computer, digital watches, etc. It is used to create digital circuits which consist of large number of logic gates.

Logic Gates:

Logic gates are building blocks of digital circuits. A logic gate performs a particular logical function. Logic gate has two or more logic inputs (LOW or HIGH) and produces a single output which may be LOW or HIGH, determined by the logic levels present at the inputs.

Q2. Explain the conversion of Boolean function to logic circuit with the help of example.

Ans: Conversion of Boolean Function to Logic Circuit:

A Boolean function can be transformed from an algebraic expression into a logic circuit composed of

A Boolean function is an expression formed with binary variables, the logical operators (OR, AND and NOT), parenthesis and equal sign. A binary variable can take the value of 0 or 1. For a given value of the variables, the function can be either 0 or 1.

Example: As an example, consider the following Boolean function.

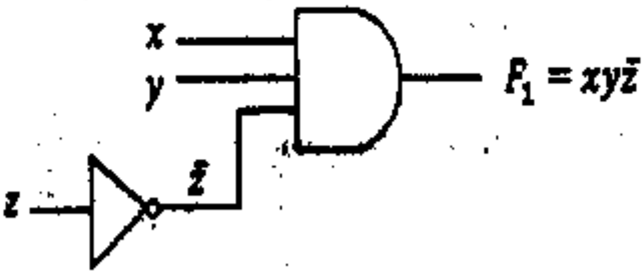
$F = x + y$

The function F is equal to 0, if $x = 0$ and $y = 0$. For all the other combinations of x and y, the function will be equal to 1.

Q3. Explain the conversion of boolean function $F_1 = xy\bar{z}$ to logic circuit.

Ans: Conversion of Boolean Function $F_1 = xy\bar{z}$ to Logic Circuit:

To convert this function to logic circuit, a single AND gate is required for the term $xy\bar{z}$. A NOT gate is also required to convert z to $\bar{z}$, before it is input to the AND gate .

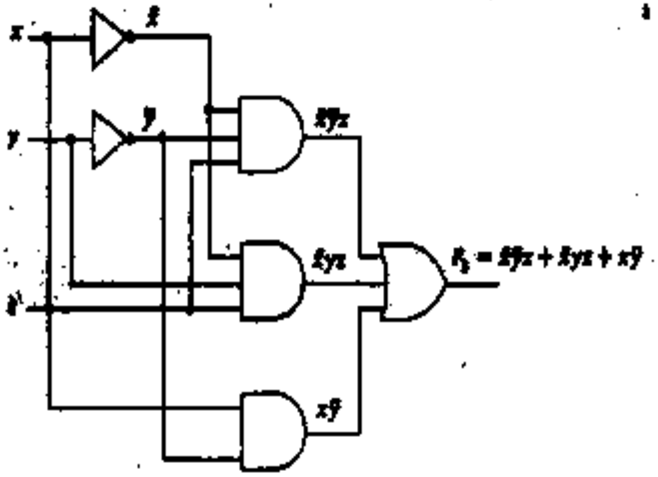


Logic circuit of Function F1

Q4. Explain the conversion of boolean function $F_3 = \bar{x}\,\bar{y}z + \bar{x}yz + x\bar{y}$ to logic circuit.

Ans: Conversion of boolean function $F_3 = \bar{x}\,\bar{y}z + \bar{x}yz + x\bar{y}$ to logic circuit:

This function has three terms. Therefore, three AND gates are required for each term. Two NOT gates are required to obtain $\bar{x}$ and $\bar{y}$. The output of AND gates is to be input into an OR gate to perform the OR operation between all the three terms.

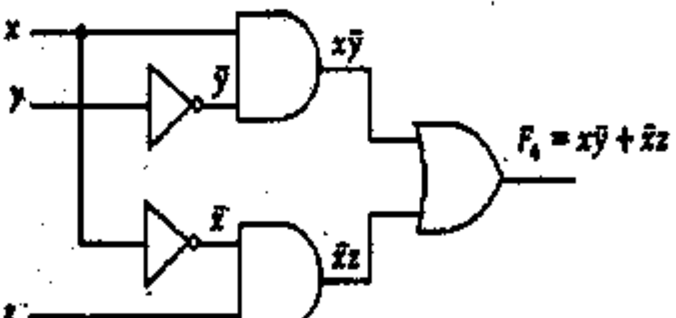


Logic circuit of function F_3

Q5. Explain the conversion of Boolean function $F_4 = x\bar{y} + \bar{x}z$ to logic circuit.

Ans: Conversion of Boolean Function $F_4 = x\bar{y} + \bar{x}z$ to Logic Circuit:

To create the logic circuit of this function, two AND gates are required for the terms $x\bar{y}$ and $\bar{x}z$, two NOT gates to convert x to $\bar{x}$ and y to $\bar{y}$ and an OR gate to perform the OR operation on the outputs of two AND gates as shown in Fig.



GUESS PAPER & MODEL PAPER # 1

Based on Chapter # 01 (Reduced Syllabus) PROGRAMMING TECHNIQUES

SECTION-A

Time allowed: 20 Minutes Marks: 12

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. Which of the following structures repeats one or more operations?
A. Sequence B. Selection
C. Loop D. Decision

ii. Which of the following structures allows a choice among various options?
A. Sequence B. Selection
C. Loop D. Decision

iii. Which of the following is a sequence of instructions written in a computer language to solve a problem?
A. Algorithm B. Flowchart
C. Program D. Problem Analysis

iv. What illustrates a sequence of operations to be performed to solve a problem in the form of a diagram?
A. Algorithm B. Flowchart
C. Program D. Problem Analysis

v. What is represented by parallelogram in a flowchart?
A. Input /Output B. Processing
C. Start/Stop D. Decision

vi. What is represented by a small circle in a flowchart?
A. Start/Stop B. Decision
C. Processing D. Connector

vii. Which symbol is used for decision in a flowchart?
A. Rectangle B. Parallelogram C. Diamond D. Oval

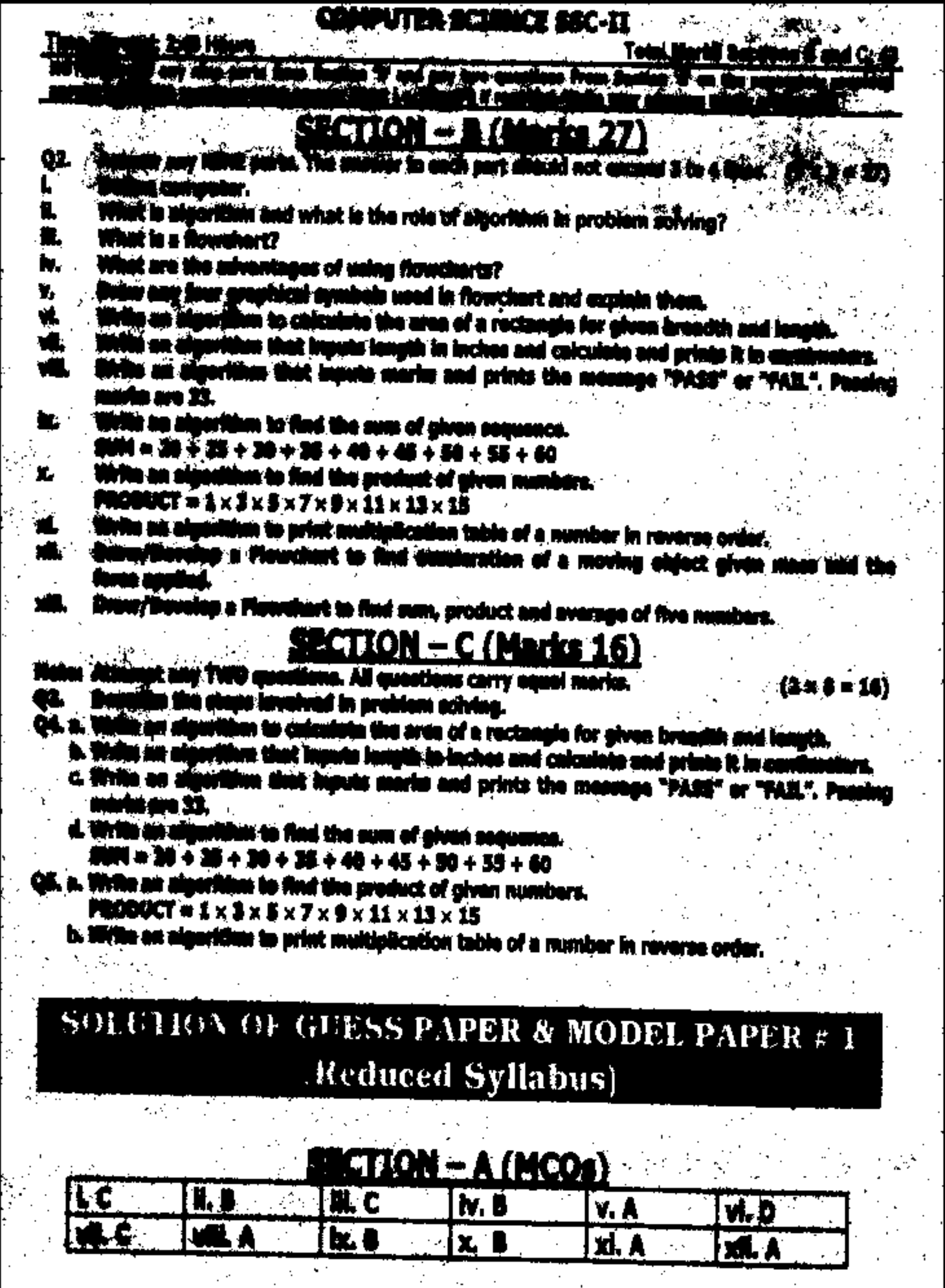
viii. Which symbol is used for processing in a flowchart?
A. Rectangle B. Parallelogram C. Diamond D. Oval

ix. ____________ allows a choice among various options while solving a problem.
A. Sequential Structure B. Selection Structure
C. Loop Structure D. Decision Structure

x. ____________ is the diagrammatic representation of algorithm.
A. Algorithm B. Flowchart
C. Program D. Problem Analysis

xi. ____________ of a problem are all the possible solutions for a problem.
A. Candid solutions B. Sequence
C. Decision D. Candid stucture

xii. ____________ refers to the execution of operations in the order in which they appear.
A. Sequential Structure B. Selection Structure
C. Loop Structure D. Decision Structure



COMPUTER SCIENCE SSC-II

Time Allowed: 2:30 Hours Total Marks Sections B and C: 43

NOTE: [illegible] any nine parts from Section 'B' and any two questions from Section 'C' on the [illegible] [illegible]

SECTION – B (Marks 27)

Q2. Attempt any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Define computer.

ii. What is algorithm and what is the role of algorithm in problem solving?

iii. What is a flowchart?

iv. What are the advantages of using flowcharts?

v. Draw any four graphical symbols used in flowchart and explain them.

vi. Write an algorithm to calculate the area of a rectangle for given breadth and length.

vii. Write an algorithm that inputs length in inches and calculate and prints it in centimeters.

viii. Write an algorithm that inputs marks and prints the message "PASS" or "FAIL". Passing marks are 33.

ix. Write an algorithm to find the sum of given sequence.
SUM = 20 + 25 + 30 + 35 + 40 + 45 + 50 + 55 + 60

x. Write an algorithm to find the product of given numbers.
PRODUCT = 1 × 3 × 5 × 7 × 9 × 11 × 13 × 15

xi. Write an algorithm to print multiplication table of a number in reverse order.

xii. Draw/Develop a Flowchart to find acceleration of a moving object given mass and the force applied.

xiii. Draw/Develop a Flowchart to find sum, product and average of five numbers.

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. Describe the steps involved in problem solving.

Q4. a. Write an algorithm to calculate the area of a rectangle for given breadth and length.

b. Write an algorithm that inputs length in inches and calculate and prints it in centimeters.

c. Write an algorithm that inputs marks and prints the message "PASS" or "FAIL". Passing marks are 33.

d. Write an algorithm to find the sum of given sequence.
SUM = 20 + 25 + 30 + 35 + 40 + 45 + 50 + 55 + 60

Q5. a. Write an algorithm to find the product of given numbers.
PRODUCT = 1 × 3 × 5 × 7 × 9 × 11 × 13 × 15

b. Write an algorithm to print multiplication table of a number in reverse order.

SOLUTION OF GUESS PAPER & MODEL PAPER # 1 (Reduced Syllabus)

SECTION – A (MCQs)

i. C	ii. B	iii. C	iv. B	v. A	vi. D
vii. C	viii. A	ix. B	x. B	xi. A	xii. A

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Define computer.

Ans: Computer:

A computer is a general-purpose electronic machine invented to help people solve various problems. Computer must be programmed by human beings to perform various tasks. Various programming techniques are used for solving problems on computer.

OR

Computer:

A computer is an electronic data processing device. It reads data processing it and produces results accurately at a very high speed.

OR

Computer is a general-purpose programmable machine that has the ability to store, retrieve and process data that is represented in the form of 0s and 1s.

OR (Second Answer)

Computer:

Computer is an advanced electronic device that takes raw data as input from the user and processes it under the control of set of instructions (called program), gives the result (output), and saves it for the future use.

ii. What is algorithm and what is the role of algorithm in problem solving?

Ans: Algorithm:

Algorithm means method, procedure, technique or plan. Algorithm is a step-by-step problem solving method that is easy to understand and follow. It is a set of steps that clearly defines a sequence of operations to solve a problem.

Role of Algorithm in Problem Solving:

Algorithm plays an important role in computer programming. Computer programming is the process of taking an algorithm and coding it in a programming language. Formulating an algorithm is the first step for developing a computer program.

iii. What is a flowchart?

Ans: Flowchart:

Flowchart is a diagrammatic representation of algorithm. It describes what operations are required to solve a given problem.

iv. What are the advantages of using flowcharts?

Ans: Importance of Flowchart in Solving a Problem:

Flowchart illustrates the sequence of operations to be performed to solve a problem in the form of a diagram.

Computer programmers draw flowcharts before writing computer programs. Flowchart provides an easy way to analyze and find solutions of problems.

Once, the flowchart is drawn, it becomes very easy to write the program in any high level language.

Flowchart is very helpful in communicating the problem solving method to other people. It also helps in finding and removing errors in computer programs.

v. Draw any four graphical symbols used in flowchart and explain them.

Ans: Flowchart Symbols:

Flowcharts are drawn using standard symbols. These symbols have specific meaning and are connected by arrows indicating the flow from one step to another.

Table 1-1 Flowchart symbols

Symbol Description	Symbol Shape	Symbol Description	Symbol Shape

Start/Stop	⬭	Decision	◇	
Input/Output	▱	Connector	○ On-Page	⬠ Off-Page

Flow Line: ⟶

It is a *line with arrow head* used to connect various flowchart symbols and indicates the flow of control in the flowchart.

Start/Stop Symbol: ⬭

It is a *rounded rectangular* shaped symbol. It is used to indicate the start or end of a flowchart. We can only write the words Start or Stop inside this symbol. A flowchart can only have one start but it may have many ends.

Input /Output Symbol: ▱

Parallelogram represents input or output operations in a flowchart. It contains the word READ or INPUT along with the variables for input operation or PRINT or OUTPUT along with the output data for output operation.

Process Symbol: ▭

A *rectangular block* is used for any data processing operation. All the calculations appear inside the processing symbol, such as "SUM= A + B". Variables are also initialized inside the process symbol, such as "K=1".

Decision Symbol: ◇

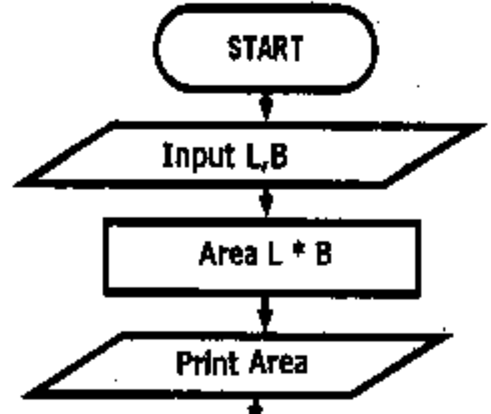
A *diamond shaped* symbol represents decision in a flowchart and it contains a condition. If the condition is true, the path marked YES is to be followed. If the condition is false, the path marked NO is to be followed. The words TRUE or FALSE can also be used instead of YES or NO.

Connector Symbols: ○ **(on page) ;** ⬠ **(off- page)**

These symbols are used to connect one part of a flowchart to the other on the same page (On-Page connector) or on the new page (Off-Page connector).

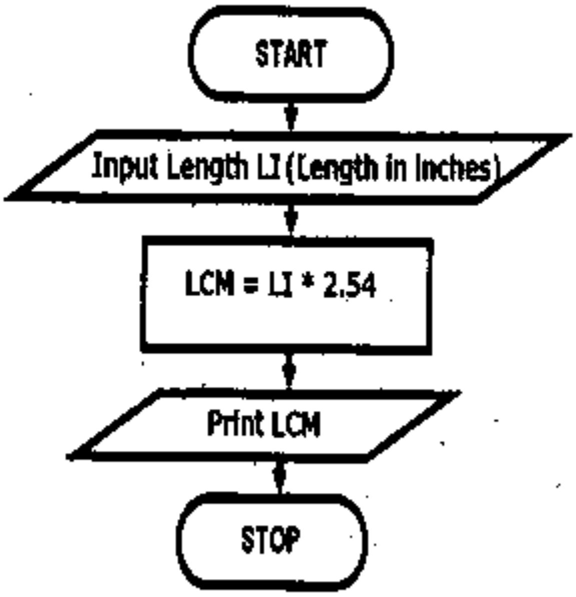
vi. Write an algorithm to calculate the area of a rectangle for given breadth and length.

Ans: Flowchart for the area of a rectangle for given breadth and length:



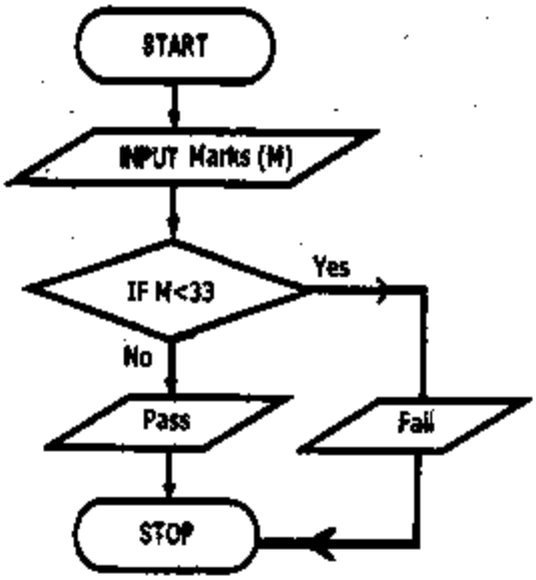
vii. Write an algorithm that inputs length in inches and calculate and prints it in centimeters.

Ans:



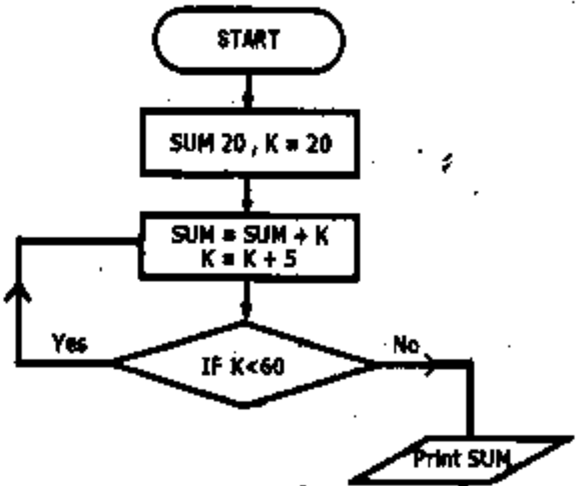
viii. Write an algorithm that inputs marks and prints the message "PASS" or "FAIL". Passing marks are 33.

Ans:



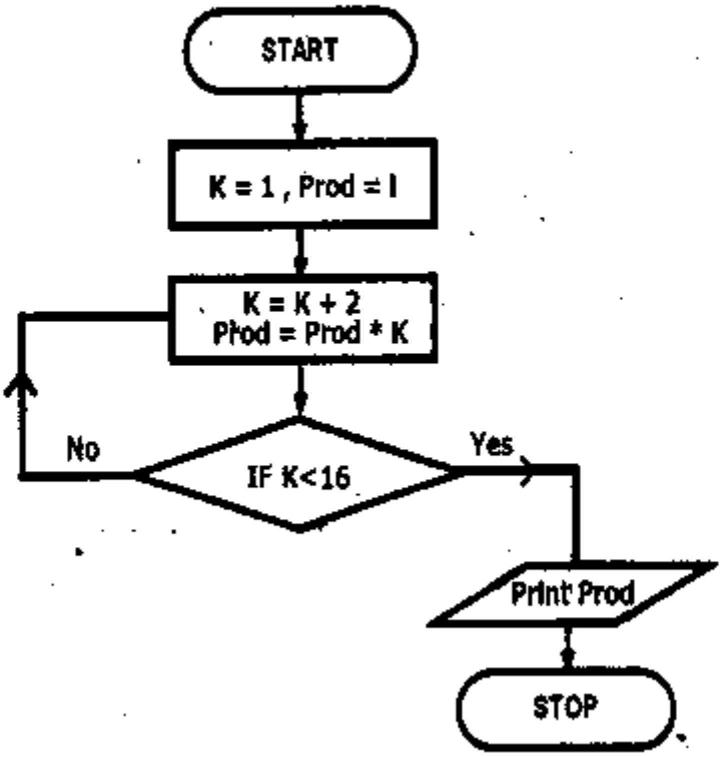
ix. Write an algorithm to find the sum of given sequence.
SUM = 20 + 25 + 30 + 35 + 40 + 45 + 50 + 55 + 60

Ans:



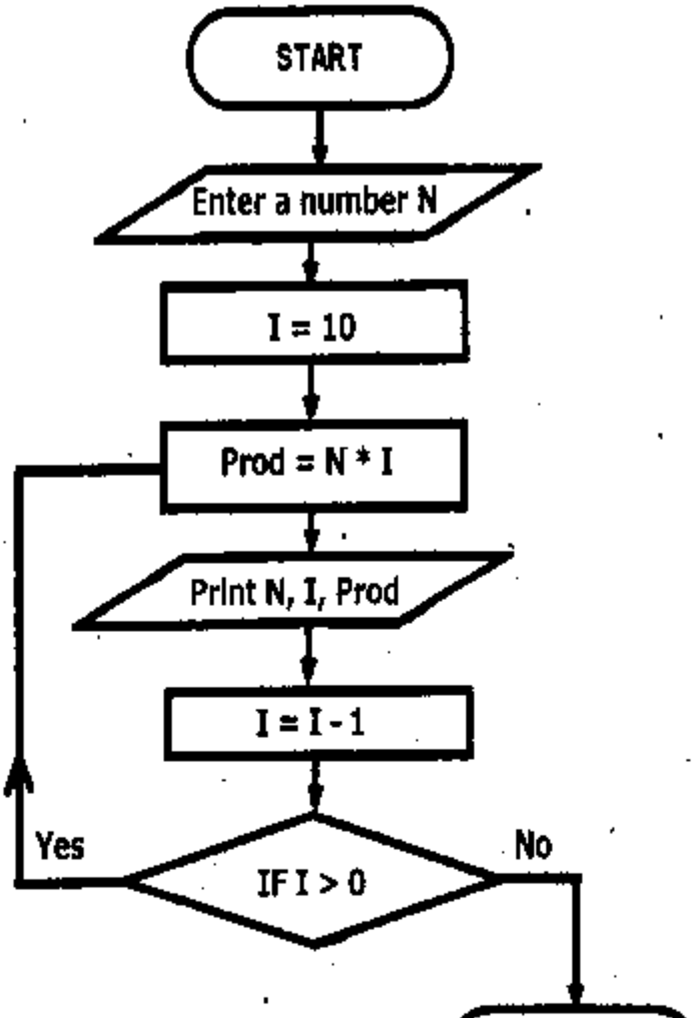
x. Write an algorithm to find the product of given numbers.
PRODUCT = 1 × 3 × 5 × 7 × 9 × 11 × 13 × 15

Ans:



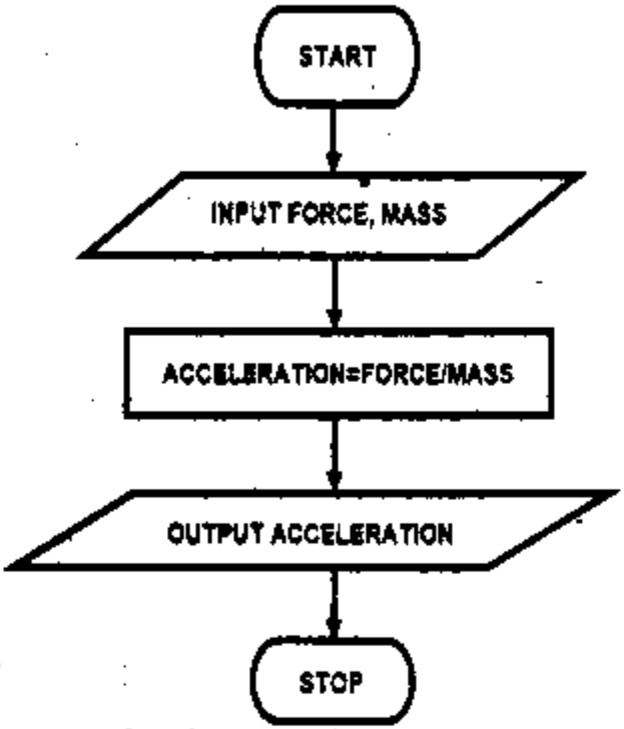
xi. Write an algorithm to print multiplication table of a number in reverse order.

Ans:



xii. Draw/Develop a Flowchart to find acceleration of a moving object given mass and the force applied.

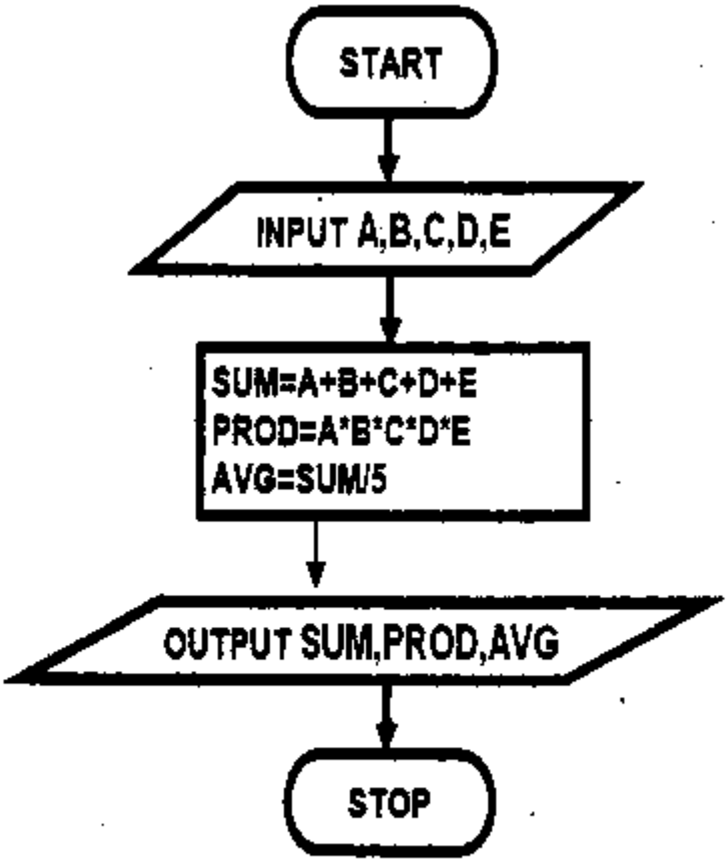
Ans:



Flowchart to find acceleration

xiii. Draw/Develop a Flowchart to find sum, product and average of five numbers.

Ans:



Flowchart to find sum, product and average

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. **(2 × 8 = 16)**

Q3. Describe the steps involved in problem solving.

Ans: The following *Five steps* are involved in problem solving on the computer.

i. Defining the Problem:

Defining the problem is initial stage of problem solving. It is very important to understand the problem before the programmer starts working on its solution.

The following are the steps to properly define and understand the problem.

- ♦ Carefully read the problem to understand what it tells.
- ♦ Find out what the problem asks to do.
- ♦ What information can be obtained from the problem?
- ♦ What is required to be calculated as the solution of the problem?

For example:

Problem 1: A Person Is Unhappy With A Product Or Service He/She Has Purchased From A Company:

After reading the above statement, we understand that a customer has purchased any product or service from a Company and now the customer is not satisfied with it. This statement defines the problem clearly and requires the solution. It is a general type of problem.

Problem 2: Finding Average Marks of a Student:

It is clear from the problem statement that average marks of a student have to be found. This is numeric problem.

ii. Analyzing The Problem:

At this stage of problem solving, the programmer investigates the problem and gathers as much information as possible to find a solution. The following questions are to be asked to analyze the problem.

- ♦ Is it possible to solve the problem on a computer?
- ♦ What is to be done to find the solution of the problem?
- ♦ What is the proper sequence of steps to solve the problem?
- ♦ What are the inputs and what output is required?
- ♦ How many solutions are possible?
- ♦ Which solution is the best and why?
- ♦ How solution will be implemented?

To analyze the **Problem 1** (given above), the following points are to be kept in mind and some information is to be gathered.

Know The Outcome You Want:

- ♦ Think about what is important to you and what you want to achieve. For example, do you want a refund, a replacement?
- ♦ Would you like them to provide the service again, this time meeting agreed standards?
- ♦ The more specific you are the more likely it is that you will get the outcome you want.

Prepare For The Discussion:

- ♦ Find out about your legal rights.
- ♦ Write down exactly what your complaint is, including details such as dates and times.
- ♦ Gather any relevant paperwork, e.g. advertisement that misled you, or a quote that the trader gave you.

Find Out What The Organisation's Complaint Process Is And Follow This:

- ♦ Have a look on their website or on any documentation, they have given you.

To analyze **Problem 2** the following information is required.

- ♦ What are the subject marks of the student in each subject?
- ♦ How many subjects are there?
- ♦ What is the formula to find the average marks?

To plan the solution of the **Problem 1** (given above), after analyzing the problem, the following planning is required:

How to Discuss a Problem:

- Talk to the right person. The person you speak must have the ability to resolve the issue, e.g. a store manager, business owner or supervisor.
- Focus on talking about the problem with the product or service, rather than taking issue with a person.
- Stay calm and reasonable. Explain the problem in detail and provide any evidence you may have.
- Tell them what outcome you want. It is the store or service provider's responsibility to resolve the problem, but it can be helpful to ask them for a specific solution.
- Expect questions. A store or service provider may ask you for more details.
- Ask to speak with someone else if you are not happy with the way the conversation is going. It is okay to walk away and come back later, or to follow up in writing.
- Listen to their response. Ask for time to consider it if you need to.

Consider Their Response:

- Does the response resolve your issue or is it a reasonable compromise?
- Do you want to take the time and effort to carry on if the response is unsatisfactory?
- Based on what you know of your rights, do you think you have a solid basis to take the matter further?

To solve the **Problem 2,** the following planning is required:

- All subject marks are to be added together to find the sum of all the marks.
- The sum is to be divided by the total number of subjects to find the average marks.

iv. Candid Solutions of a Problem:

All the possible solutions of a problem that produce correct result are known as candid solutions. To find candid solutions of a problem, programmer has to look for different methods to solve the problem and come up with several solutions.

For example, many solutions can be considered to resolve the **Problem 1**.

Solution 1: The customer can lodge the complaint personally visiting the company.

Solution 2: The customer can also use the courier services to send the product back.

Solution 3: The customer can use company's website to file the complaint.

v. Select The Best Solution:

After finding the candid solutions, only one solution can be selected. The selection of final solution of a problem should be based on the following criteria.

Speed:

The selected solution of the problem should be efficient. In other words, it means when the solution is implemented in a programming language, the program should run fast.

Cost: The selected solution of the problem should provide a cost-effective way of implementation.

Complexity:

The selected solution of the problem should not be complicated. It should contain minimum number of instructions/simple steps.

Q4. a. Write an algorithm to calculate the area of a rectangle for given breadth and length.

Ans:
```
Step 1:   Input the width (W) and Length (L) of a rectangle
Step 2:   Calculate the area (A) by multiplying L with W
Step 3:   Print A
```

b. Write an algorithm that inputs length in inches and calculate and prints it in centimeters.

Ans:
```
Step 1:   Input the length in inches (LI)
Step 2:   Calculate the length in cm (LCM) by multiplying LI with 2.54
Step 3:   Print LCM
```

c. Write an algorithm that inputs marks and prints the message "PASS" or "FAIL". Passing marks are 33.

Ans:
```
Step 1:   Input Marks (M)
```

d. Write an algorithm to find the sum of given sequence.
SUM = 20 + 25 + 30 + 35 + 40 + 45 + 50 + 55 + 60

Ans:

```
Step 1: Start
     Initialize SUM to 0 and K = 5
               SUM=0, K=5
Step 2: ADD K to SUM
               SUM=SUM+K
Step 3; Increment K by 5
               K=K+5
Step 4: Check if the value of K is less than or equal to 60
        IF K≤100 THEN GOTO Step 2 otherwise GOTO Step 5
Step 5: Print SUM
Step 6: Stop
```

Q5. a, Write an algorithm to find the product of given numbers.
PRODUCT = 1 × 3 × 5 × 7 × 9 × 11 × 13 × 15

Ans:

```
Step 1: Start
     Initialize variable K to 1 and prod = 1
               K=1   Prod= 1
Step 2: Increment K by 2
               K=K+2
Step 3:  Find the Product
        Prod = Prod × k

Step 4: Check if the value of K is less than 16
        IF K<16 THEN GOTO Step 2 otherwise GOTO Step 5
Step 5: Print product
        Print Prod
Step 6: Stop
```

b. Write an algorithm to print multiplication table of a number in reverse order.

Ans:

```
Step 1: Enter the number N Whose table is to be generated
Step 2: initialize the valve of I with 10
        I = 10
Step 3: Find the product of N and I
        Prod = N x I
Step 4: Print N, I and Prod
        Print  N    I  and Prod
Step 5: Decrease the value of I by 1
        I = I – 1
Step 5: If the value of I is > 0 then goto step 3 Other wise goto step 6
Step 6: Stop
```

IMPORTANT QUESTIONS & ANSWERS

Q1. Explain the criteria for measuring efficiency of an algorithm.

Ans: Measuring Efficiency of an Algorithm:

The efficiency of an algorithm is the property which retate the algorithm to the amount of computational resources used in it. An algorithm must be analyzed to determine its resource usage (e.g. time, memory and storage space).

For maximum efficiency we wish to minimize resource usage. However, the various resources cannot be compared directly, so which of two algorithms is considered to be more efficient often depends on which measure of efficiency is considered the most important, e.g. the requirement for high speed, minimum memory usage is considered.

There may be several algorithms for solving a particular problem. How fast a problem can be solved, depends on efficient algorithm.

Therefore, *analysis of algorithms* is very important to compare algorithms and conclude that which one is better than the other.

Analyzing the efficiency of algorithms means comparing efficiency of different methods of solutions of a problem.

Q2. Problem: Find the Sum, Product and Average of Five given Numbers.

Ans: Planning the Solution:

Input: Five given numbers
Required Output: Sum, product and average of five numbers
Processing: Addition, multiplication and division of numbers

Algorithm:

```
Step 1: Start
        Let the five numbers be A=2, B=5, C=8, D=4 and E=12
Step 2: FIND the sum (SUM)
                SUM=A+B+C+D+E
Step 3: FIND the product (PROD)
                PROD=A*B*C*D*E
Step 4: FIND the average (AVG)
                AVG=SUM/5
Step 5: Output SUM, PROD, AVG
Step 6: Stop
```

Q3. Problem: Find the Largest of Three Unequal Numbers.

Ans: Planning the Solution:

Input: Three unequal numbers
Required Output: The largest of three numbers
Processing: Comparison of each number with the other two one by one

Algorithm:

```
Step 1: Start
        Let the three numbers be A=10, B=20 and C=30
Step 2: Check if A is the largest number
        IF A>B and A>C THEN LARGEST=A otherwise GOTO Step 4
Step 3: GOTO Step 7
Step 4: Check if B is the largest number
        IF B>A and B>C THEN LARGEST=B otherwise GOTO Step 6
Step 5: GOTO Step 7
Step 6: LARGEST=C
```

Q4. Problem: Find Acceleration of a Moving Object for Given Mass and the Force Applied.

Ans: Planning the Solution:

Input: Mass and the Force
Required Output: Acceleration of moving object
Processing: Divide the force by the mass

Algorithm:

```
Step 1: Start
Let the mass (M) be 50 and the force (F) be 12
Step 2: CALCULATE the acceleration (A)
        A = F/M
Step 3: Output A
Step 4: Stop
```

Q5. Problem: Find the Volume of a Cube.

Ans: Planning the Solution:

Input: Length of side of cube
Required Output: Volume of cube
Processing: Side of cube raised to the power of three

Algorithm:

```
Step 1: Start
    Let the length of one side of cube, L be 12
Step 2: CALCULATE the volume, V
        $V = L^3$
Step 3: Output V
Step 4: Stop
```

Q6. Problem: Find the Area of a Triangle when the Lengths of Height and Base are given.

Ans: Planning the Solution:

Input: Height and base of triangle
Required Output: Area of triangle
Processing: Multiply height with base and divide by two

Algorithm:

```
Step 1: Start
   Let the height, H be 10 and the base, B be 15
Step 2: CALCULATE the area, A
        A = (B*H)/2
Step 3: Output A
Step 4: Stop
```

Q7. Problem: Read Marks (M) and Print Letter Grade According to the Following Scheme:

Marks	Letter Grade
M≥80 and M≤100	A1
M≥70 and M<80	A
M≥60 and M<70	B
M≥50 and M<60	C
M≥40 and M<50	D
M≥33 and M<40	E
M≥0 and M<33	Ungraded

Ans: Planning the Solution:

Input: Marks

```
        Let the marks, be M
    Step 2: Compare the marks
        IF M≥80 THEN print "Grade A1" otherwise GOTO Step 4
    Step 3: GOTO Step 13
    Step 4: Compare the marks
        IF M≥70 THEN print "Grade A" otherwise GOTO Step 6
    Step 5: GOTO Step 13
    Step 6: Compare the marks
        IF M≥60 THEN print "Grade B" otherwise GOTO Step 8
    Step 7: GOTO Step 13
    Step 8: Compare the marks
        IF M≥50 THEN print "Grade C" otherwise GOTO Step 10
    Step 9: GOTO Step 13
    Step 10: Compare the marks
        IF M≥40 THEN print "Grade D" otherwise GOTO Step 12
    Step 11: GOTO Step 13
    Step 12: Compare the marks
        IF M≥33THEN print "Grade E" otherwise print "Ungraded"
    Step 13: Stop
```

Q8. Problem: Find the Exponent of a Given Number:

Ans: Exponent or power of a number means how many times to use the number in a multiplication. In other words it is the product of a number that is multiplied as many times as its exponent.

Planning the Solution:

Input: A number and its exponent
Required Output: Exponent of given number
Processing: Multiply the number as many times as its exponent

Algorithm:

```
    Step 1: Start
    Let the number, N be 8 and its exponent, E be 5
    Step 2: Initialize product(P) and K to 1
            P=1, K=1
    Step 3: FIND the product(P)
            P=P*N
    Step 4: Increment K by 1
            K=K+1
    Step 5: Check if the value of K is less than or equal to E
        IF K≤E THENGOTO Step 3 otherwise GOTO Step 6
    Step 6: Output P
    Step 7: Stop
```

Q9. Problem: Print Odd Numbers from 1 to 100:

1 3 5 7 9 11 . . . 99

Ans: Planning the Solution:

Input: This problem has no input
Required Output: Printing odd numbers from 1 to 100
Processing: Initialize a variable to 1 and keep printing it with an increment of 2 till 99

Algorithm:

```
    Step 1: Start
        Initialize variable K to 1
        IF K<100 THEN GOTO Step 2 otherwise GOTO Step 5
Step 5: Stop
```

Q10. Problem: Print The Following Sequence Of Numbers In Descending Order:
27 24 21 18 15 12 9 6 3 0 -3 -6

Ans: Planning the Solution:

Input: This problem has no input

Required Output: Printing numbers from 27 to -6 in descending order with a step of -3

Processing: Initialize a variable to 27 and then keep printing it with a decrement of 3 till -6

Algorithm:

```
Step 1: Start
    Initialize variable K to 27
            K=27
Step 2: Output K
Step 3: Decrement K by 3
            K=K-3
Step 4: Check if the value of K is greater than or equal to -6
        IF K≥-6 THEN GOTO Step 2 otherwise GOTO Step 5
Step 5: Stop
```

Q11. Problem: Find the Sum of Even Numbers up to 100:
SUM = 2 + 4 + 6 + 8 + 10 + 12 + 14 + . . . + 100

Ans: Planning the Solution:

Input: This problem has no input

Required Output: Sum of even numbers up to 100

Processing: Add all the even numbers from 2 to 100

Algorithm:

```
Step 1: Start
    Initialize SUM to 0 and K to 2
            SUM=0, K=2
Step 2: ADD K to SUM
            SUM=SUM+K
Step 3: Increment K by 2
            K=K+2
Step 4: Check if the value of K is less than or equal to 100
        IF K≤100 THEN GOTO Step 2 otherwise GOTO Step 5
Step 5: Output SUM
Step 6: Stop
```

Q12. Problem: Print a Multiplication Table of a Given Number:

Ans: Planning the Solution:

Input: Number whose table is required

Required Output: Printing table of given number

Processing: Initialize a variable to 1 and print its product with the given number. Increment the variable by 1 and print the product. Continue the process till printing the product of 12 and the given number.

Algorithm:

```
Step 1: Start
    Let the number, N be 7
Step 2: Initialize K to 1
            K=1
Step 3: FIND the product
```

```
    Step 6: Check the value of K
        IF K≤12 THEN GOTO Step 3 otherwise GOTO Step 7
    Step 7: Stop
```

Q13. Problem: Convert Temperature from Fahrenheit to Celsius:

Ans: Planning the Solution:

Input: Temperature in Fahrenheit

Required Output: Temperature in Celsius

Processing: Compute the temperature in Celsius from Fahrenheit using conversion formula

Algorithm:

```
Step 1: Start
    Let the temperature in Fahrenheit, F be 100
Step 2: CALCULATE temperature in Celsius (C)
        C = 5/9(F-32)
Step 3: Output C
Step 4: Stop
```

Q14. Problem: Find Factorial of a Given Number:

Ans: Planning the Solution:

Input: Given Number

Required Output: Factorial of given number

Processing: Find the product of all the numbers from 1 to the given number.

Algorithm:

```
Step 1: Start
    Let the number, N be 5
Step 2: Initialize loop variable K and factorial F to 1
        K=1, F=1
Step 3: CALCULATE the product
        F=F*K
Step 4: Increment K by 1
        K=K+1
Step 5: Check the value of K
    IF K≤N THEN GOTO Step 3 otherwise GOTO Step 6
Step 6: Output F
Step 7: Stop
```

Q15. Determine the flowchart requirements of a given problem or algorithm by the flowchart developer before drawing a flowchart.

Ans: Steps for Drawing Flowchart:

The flowchart developer must determine the following requirements for the given problem or algorithm before drawing a flowchart.

- Input to the flowchart
- Type of processing required
- Decisions to be taken
- The output to be produced after processing

Input:

The flowchart developer must know what exactly the input to the flowchart is. The input is determined from the problem statement. For example, the given problem is to convert temperature from Fahrenheit to Celsius. Here, the input will be the temperature in Fahrenheit.

Processing:

The flowchart developer must decide what type of calculation is to be performed or which formula is to be applied to obtain the required result. For example, to find the area of a triangle, the following formula is to

Decision:

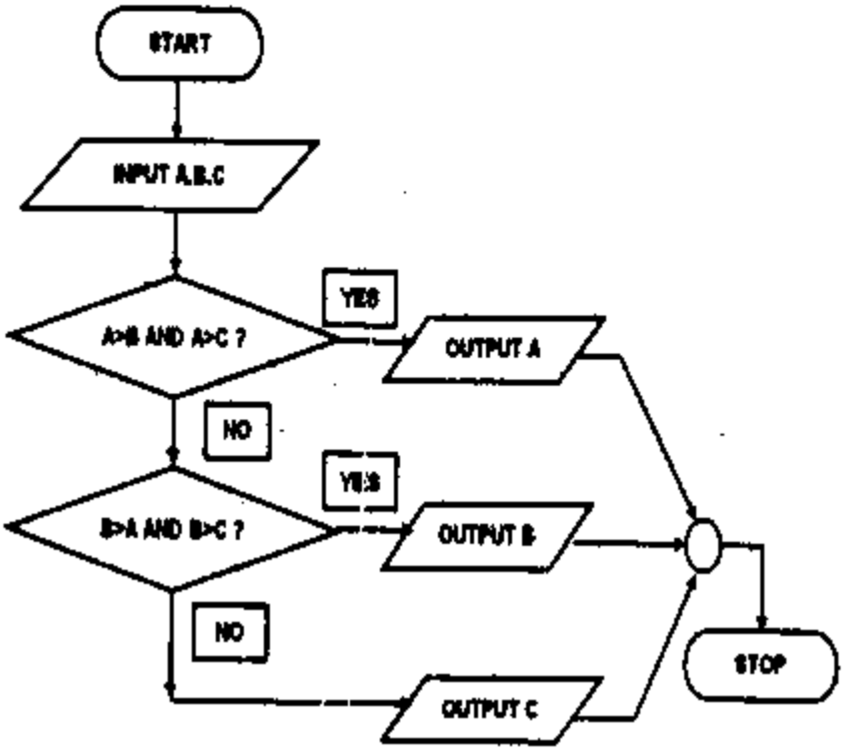
The flowchart developer must decide which control structures (sequence, repetition or selection) are to be applied for the solution of the problem. For example, selection structure must be applied to print letter grade of a student based on the marks obtained. The selection structure will check in which range the marks fall and accordingly print the grade.

Output:

The flowchart must provide the required output.

Q16. Draw/Develop a Flowchart to find the largest of three unequal numbers.

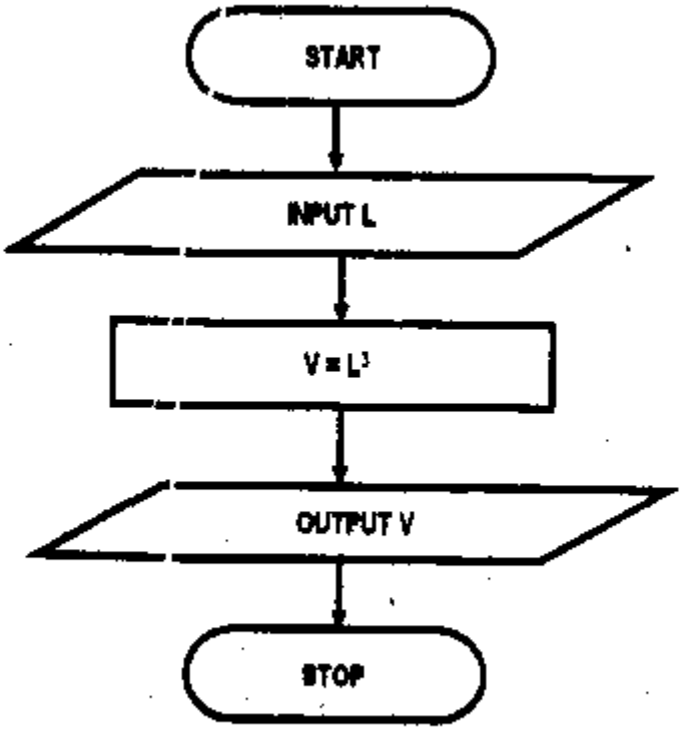
Ans:



Flowchart to find largest of three numbers

Q17. Draw/Develop a Flowchart to input the length of one side of cube and print its volume.

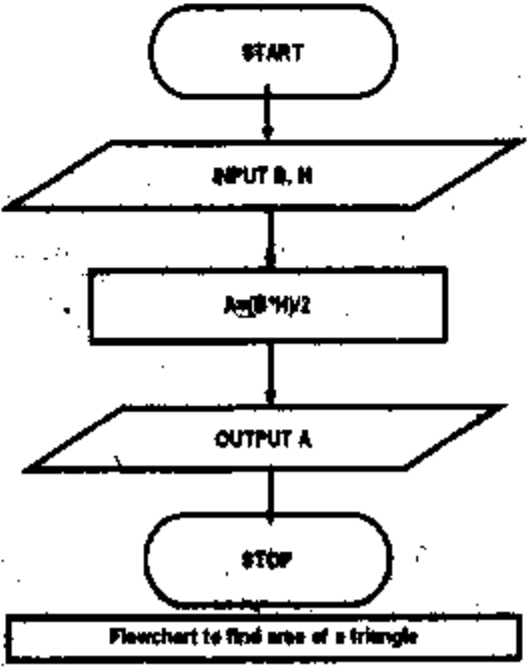
Ans:



Fl owchart to find volume of a cube

Q18. Draw/Develop a Flowchart to find the area of a triangle when the lengths of height and base are given.

Ans:

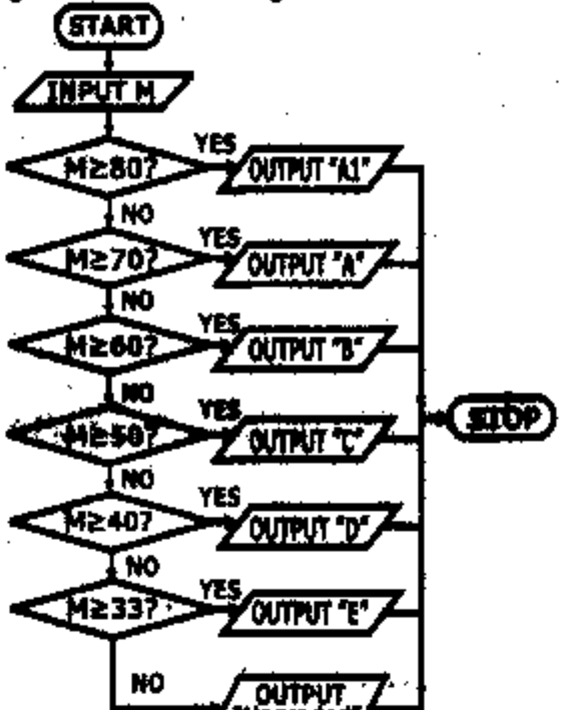


Flowchart to find area of a triangle

Q19: Draw/Develop a Flowchart to read marks (M) and print letter grade according to the following criteria

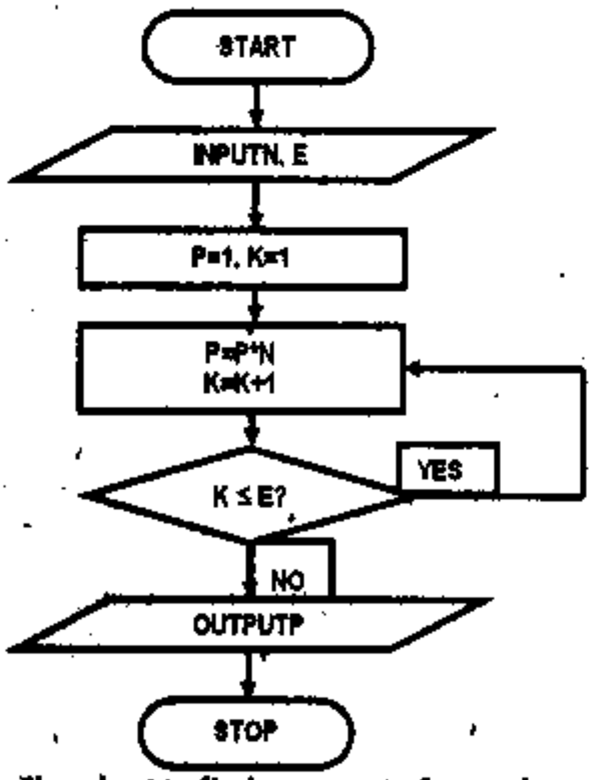
Marks	Letter Grade
M≥80 and M≤100	A1
M≥70 and M<80	A
M≥60 and M<70	B
M≥50 and M<60	C
M≥40 and M<50	D
M≥33 and M<40	E
M<33	Ungraded

Ans: Flowchart to print letter grade is shown in Figure.



Q20. **Draw/Develop a Flowchart to find the exponent (E) of a given number (N) In this flowchart, the value P represents N raised to the power E.**

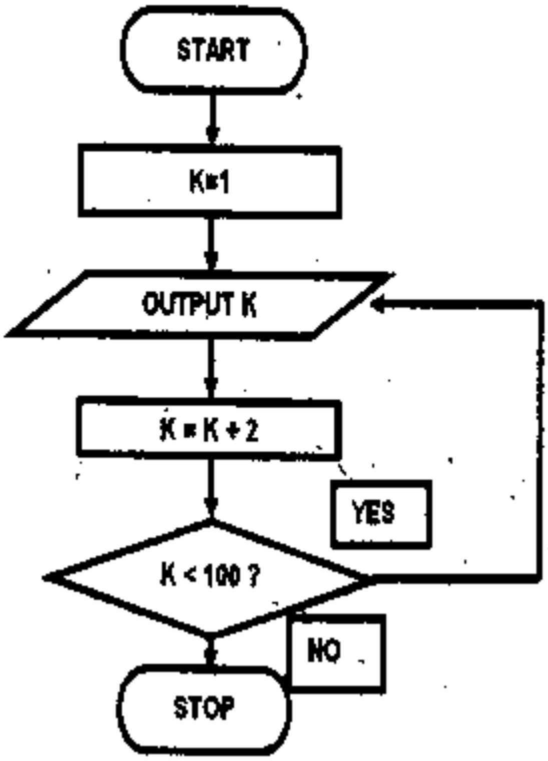
Ans:



Flowchart to find exponent of a number

Q21. **Draw/Develop a Flowchart to print odd numbers from 1 to 100**

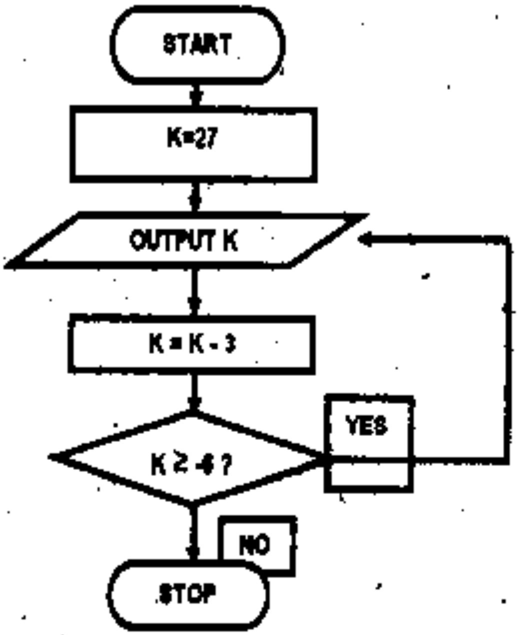
Ans:



Flowchart to print odd numbers

Q22. **Draw/Develop a Flowchart to print the given sequence of numbers in descending order. 27 24 21 18 15 12 9 6 3 0 -3 -6**

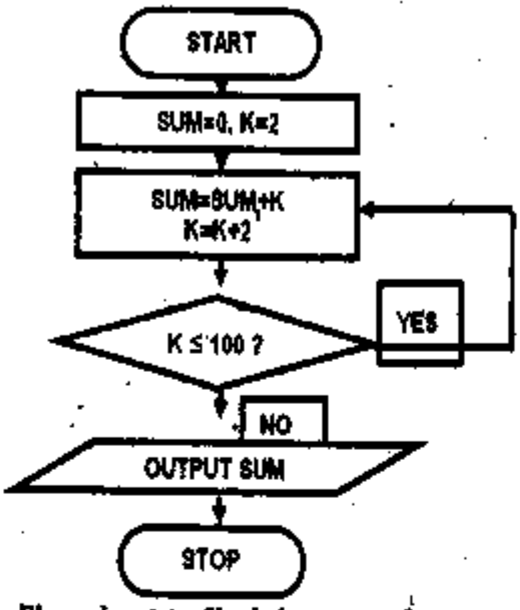
Ans:



Flowchart to print numbers in descending order

Q23. **Draw/Develop a Flowchart to find the sum of even numbers up to 100.**

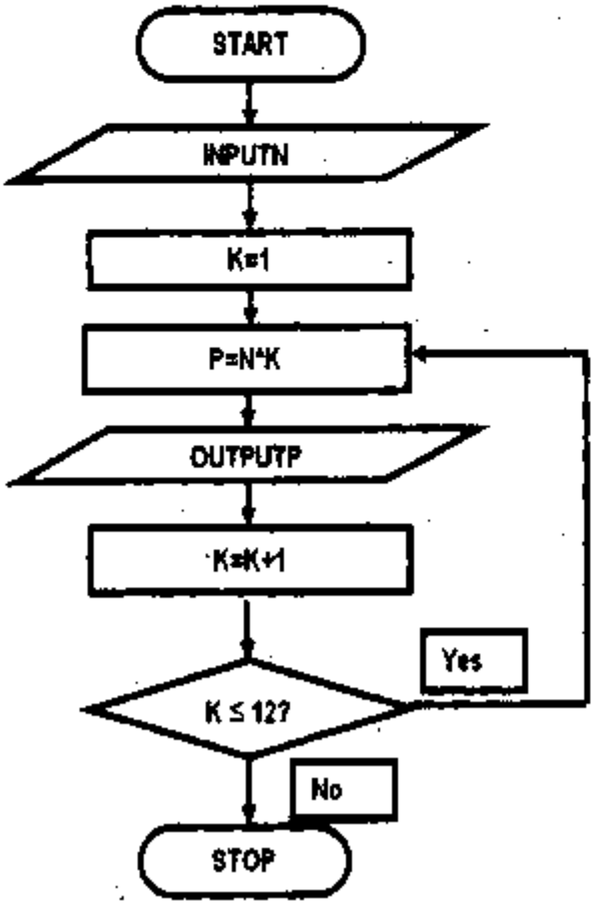
Ans:



Flowchart to find the sum of even numbers

Q24. Draw/Develop a Flowchart to print a multiplication table of a given number.

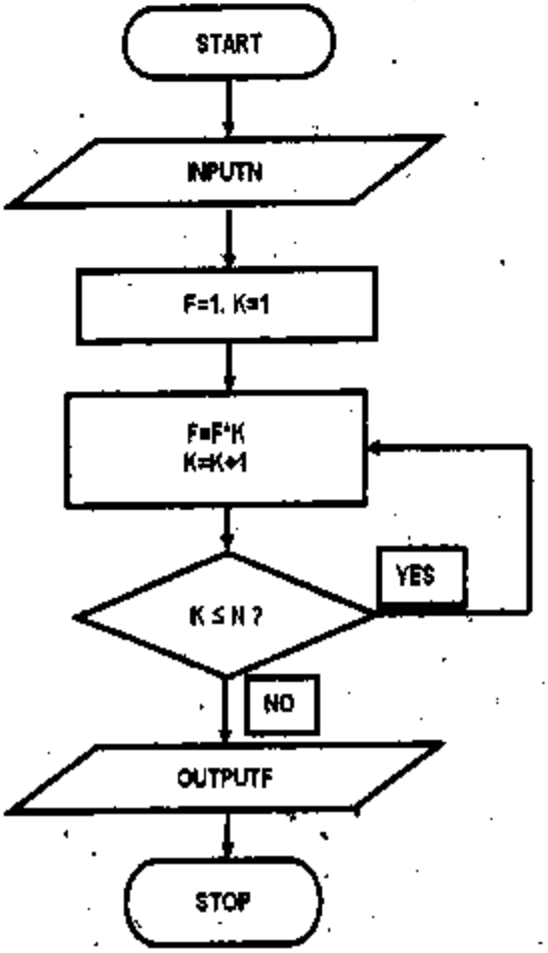
Ans:



Flowchart to print multiplication table of a number

Q25. Draw/Develop a Flowchart to find factorial of a number.

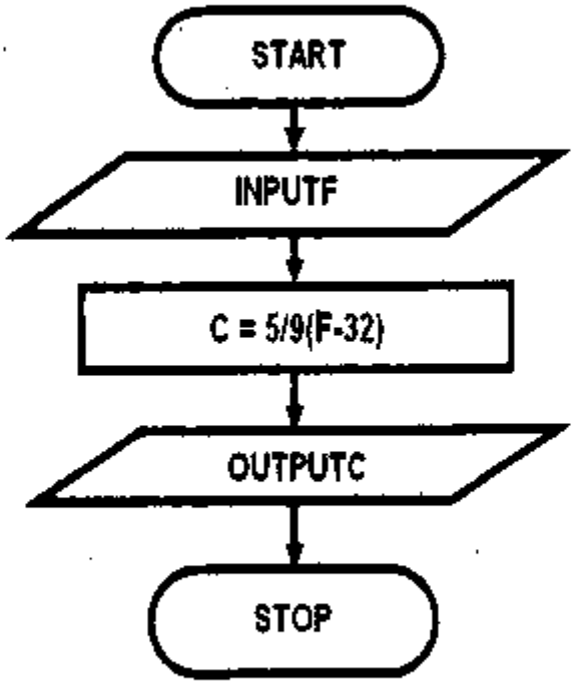
Ans:



Flowchart to print factorial of a number

Q26. Draw/Develop a Flowchart to convert temperature from Fahrenheit to Celsius.

Ans:



Flowchart to convert temperature

GUESS PAPER & MODEL PAPER # 2

Based on Chapter # 02 (Reduced Syllabus) PROGRAMMING IN C

SECTION-A (Marks 12)

Time allowed: 20 Minutes Marks: 12

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. What defines the rules of valid statements in programming?
A. Compiler B. Interpreter
C. Syntax D. Semantic

ii. Which language is directly understood by the computer?
A. Machine language B. Assembly language
C. High level language D. C language

iii. When was C language developed?
A. Late 1960s B. Early 1970s C. 1980s D. 1990s

iv. Who developed Java language?
A. Dennis Ritchie B. Microsoft C. Sun Microsystems D. IBM

v. What is the other word used for Reserved Words?
A. Compiler words B. Keywords
C. Special programming words D. Mnemonics

vi. How many bytes are set aside by the compiler for a variable of type 'int'?
A. 2 B. 3 C. 4 D. 5

vii. How many bytes are set aside by the compiler for a variable of type float?
A. 2 B. 3 C. 4 D. 5

viii. What is the range of numbers that can be stored in a variable of type double float?
A. -32,768 ~ +32,767 B. 10^{-38} ~ 10^{38}
C. 10^{-308} ~ 10^{308} D. 10^{-4932} ~ 10^{4932}

ix. Which program translates high level language into machine language?
A. Compiler B. Linker
C. Loader D. Debugger

x. Which software helps in finding and removing errors in programs?
A. Linker B. Text Editor
C. Loader D. Debugger

xi. A quantity whose value does not change during program execution is called:
A. Constant B. Linker
C. Loader D. Debugger

xii. Which of the following functions all C-programs MUST contain?
A. main() B. system() C. scanf() D. getche()

SECTION – B (Marks 27)

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Define computer program.
ii. Differentiate between syntax and semantic.
iii. Write three differences between assembly language and HLLs.
iv. Write four characteristics of HLLs.
v. Define Integrated Development Environment (IDE).
vi. Differentiate between constant and variable.
vii. Which of the following are valid C variables? Give the reason if not a valid variable area,
5x, Sum, net pay, float, _age, else, case, size22, my_weight
viii. What are reserved words? Why they should not be used as variable names?
ix. Why comments are used in programs?
x. What is the purpose of header files in C language?
xi. What are the rules for specifying a variable name in C language?
xii. Differentiate between compiler and interpreter.
xiii. Explain The main() Function.

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. Describe the following High Level Languages.

a) C/C++ b) Visual Basic
c) C# d) Java

Q4. What is C language IDE? Explain its modules in detail.

Q5. What is a preprocessor directive? Explain Include# preprocessor directive in detail.

SOLUTION OF GUESS PAPER & MODEL PAPER # 2
(Reduced Syllabus)

SECTION- A (MCQs)

i. C	ii. A	iii. B	iv. C	v. B	vi. A
vii. C	viii. C	ix. A	x. D	xi. A	xii. A

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

I. Define computer program.

Ans: Computer Program:

A computer program is a set of instructions (statements) written in a programming language to solve a particular problem and achieving specific results. Any task performed by a computer is controlled by a set of instructions that are executed by the microprocessor. A large variety of programming languages have been developed for writing computer programs to use the computer as a problem-solving tool. Each statement of a programming language has syntax and semantic.

ii. Differentiate between syntax and semantic.

Ans: ♦ Syntax:

Syntax refers to the rules of a programming language according to which statements of a program are to be written. It describes the way to write correct statements in a program. Syntax of a programming language is similar to the grammar of a natural language.

For example, an assignment statement consists of a variable and an expression separated by equal sign as

- **Semantic:**

Semantic gives meaning to statements of a programming language. It describes the sequence of operations to be performed by a computer when executing the statements of a computer program.

For example, in the assignment statement:

sum = a + b;

the semantic of the statement is to perform the expression, that is, add the values stored in variables a and b and then store the result in variable sum.

iii. Write three differences between assembly language and HLLs.

Ans: Differences between high level and assembly language:

Features of high level language (HLLs):

1. High level languages are easily understandable.
2. The programs that are developed in high level language are portable.
3. In case of high level languages debugging of the code is easy and the program written is not machine dependent.

Features of Assembly language:

1. Although Assembly level languages are not easy to understand they are relatively easier as compared to machine level languages.
2. The programs written in this language are not portable and the debugging process is also not very easy.
3. The programs developed in assembly language are thoroughly machine dependent.

iv. Write four characteristics of HLLs.

Ans: Characteristics of High Level Languages:

High-level languages have the following characteristics.

1) These languages were developed to make computer programming simple, easier and less prone to errors.
2) High level languages are not machine dependent. They enable programmers to write programs that are independent of a particular type of computer.
3) Programs written in high-level languages must be translated into machine language by a compiler or an interpreter before execution by the computer.
4) The process of finding and removing errors in programs (debugging) is easier in high-level languages compared to low level language.
5) High-level language programs are highly structured. They allow programmers to break lengthy programs into a number of modules which can be written and tested independently. This makes writing and testing of programs easier.

v. Define Integrated Development Environment (IDE).

Ans: Integrated Development Environment (IDE):

IDE is computer software that brings all the processes and tools required for program development into one place. IDE's aim is to make the life of programmers easier by grouping together all the tasks needed for building applications into one environment.

Today's modem IDEs have user-friendly Graphical User Interface (GUI). Most of the new programming languages use Integrated Development Environment (IDE) to create, compile and run programs.

vi. Differentiate between constant and variable.

Ans: Constants and Variables:

Constant and variables are used in expressions in computer programs. After an expression is evaluated, the result of the expression is also stored in a variable.

- **Constant:**

Constants are quantities whose values do not change during program execution.

Types of Constant:

They may be numeric, character or string.

Integer constants represent values that are counted, like the number of students in a class. Some examples of integer constants are 7145, -234, 28, etc.

Floating-point constants are used to represent values that are measured, like the height of a person which might have a value of 166.75 cm or the weight such as 82.6 kilograms.

2. Character Constant:

Character Constant is one of the symbols in C character set. It includes digits 0 to 9, upper-case letters A to Z, lower-case letters a to z, punctuation symbols such as semicolon (;), comma (,), period (.) and special symbols such as +, -, =, >, etc. A character constant is enclosed by single quotes such as 'a','s', etc.

String Constant:

String Constant contains a string of characters within double quotes such as "Hello Ahmed", "a", etc.

♦ **Variable:**

A variable is a symbolic name that represents a value that can change during execution of a program. A variable has a name, known as variable name and it holds data of other types. A number or any other type of data held in a variable is called its value. It also indicates the types of value a variable can represent.

Types of Variables:

Variables are of two types, numeric and character.

1. Numeric variables:

Numeric variables are used to represent numeric values in computer programs. They represent integer and floating-point values. Some examples of numeric variables are sum, average, length, salary, marks, etc.

2. Character variables:

Character variables represent character values in computer programs. It can represent a single character or a string of characters. Some examples of character variables are name, city, gender, etc.

When a variable is used in a computer program, the computer associates it with a particular memory location. The value of a variable at any time is the value stored in the associated memory location at that time. Variables are used so that the same space in memory can hold different values at different times.

vii. Which of the following are valid C variables? Give the reason if not a valid variable area, 5x, Sum, net pay, float, _age, else, case, size22, my_weight

Ans: Valid: my_weight, area, size22, Sum

Not Vaild:

5x	**A variable can't be start with number**
net pay	**Space can't be used in variable name**
float	**Data types can't be used as variable name**
_age	**Special character (_) Cant be used in variable name**
else	**Commands cannot be used as variable name**
case	**Commands cannot be used as variable name**

Reason: **Space not allowed in variable names.**
Special characters and numbers are not allowed start the name of variable.
Reserve words are not allowed.

viii. What are reserved words? Why they should not be used as variable names?

Ans: Reserved Words:

The words that are part of programming language and have special purposes in computer programs are called reserved words or keywords.

Reserved words have predefined use and *cannot be used* for any other purpose. Reserved words are always written in lowercase. There are 32 words defined as reserved words in C. A complete list of reserved words used in C language is given in table.

List of reserved words of C:

Reserved Words

case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

ix. Why comments are used in programs?

Ans: Comments in C Language:

It is a good programming practice to add comments in program to make it easy for others to understand it. Comments in the source code are ignored by the compiler.

Comments are added in programs when a fact is necessary to be brought to the attention of program's reader. Generally, programmers write comments at the beginning of the program explaining the reader what the program is intended to achieve and inside the program code where something is to be clarified about the structure of the program.

There are two types of comments.

- Single line comment
- Multiple line comment

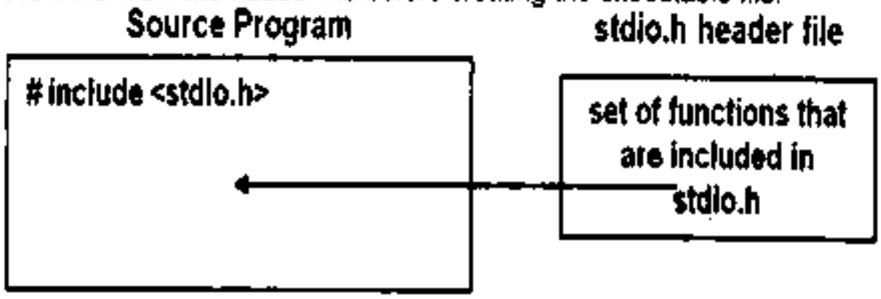
x. What is the purpose of header files in C language?

Ans: Purpose of header files in C language:

Some commonly used header files are stdio.h, conio.h and math.h.

- Some commonly used functions that are included in stdio.h file are printf(), scanf(), etc.
- Some of the functions that are included in conio.h file are clrscr(), getch() and getche().
- The math.h header file contains mathematical function such as sqrt(), pow(), log(), sin(), cos() and tan().

When source program is translated into executable file, a copy of code of function is pasted in the source program by the linker from the header file before creating the executable file.



Use of header file in a C program

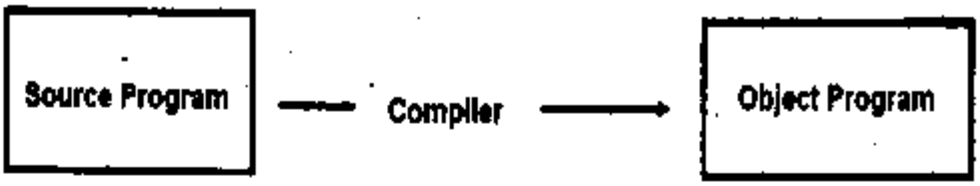
xi. What are the rules for specifying a variable name in C language?

Ans: Rules for Specifying Variable Names in C language:

The following are the rules for specifying variable names in C language.

- A variable begins with a letter or underscore (_) and may consist of letters, underscores and/or digits.
- The underscore may be used to improve readability of the variable name. For example, **over_time**.
- There is no restriction on the length of a variable name. However, only the first 31 characters of a variable are significant. This means that if two variables have the same first 31 characters they are considered to be the same variables.
- Both upper and lower case letters are allowed in naming variables. An upper case letter is considered different from a lower case letter. For example the variable AVG is different from Avg or avg.
- Special characters cannot be used as variable name. e.g., #, ?, @ etc.
- Reserved words of C language such as **int**, **case**, If, etc., cannot be used as variable names.
- Space is not allowed in the name of variable. For example **ma ss** is not correct.

xii. Differentiate between compiler and interpreter.



Translation of source program into object program

Source program consists of statements written in a high level language such as C, Pascal, Java, etc.

For example, a program written in C language by a programmer to print table of a number is known as source program. When it is translated with a compiler into machine language, the resulting program is known as object program.

The object program is understandable by computer processor but difficult for a human to read and understand because it consists of zeroes and ones.

Interpreter:

Interpreter translates high level language programs into machine language but it translates one instruction at a time and executes it immediately before translating the next instruction.

Examples of programming languages that use interpreter are Java Script, BASIC, Visual Basic and Perl.

Interpreter reads each statement of source program, one at a time and determines what it means as it executes it. It means each time a statement is read, it must be translated into machine language before execution. Compiler translates the entire program into object program before execution by computer. Therefore, a compiled program runs fast.

xiii. Explain The main() Function.

Ans: The main() Function:

C program consists of one or more functions. A function performs a single well-defined task. Every C program must have the function **main()** which is the first section to be executed when the program runs. The general form of main() is:

Void main(void)

The word **void** before the function **main()** means that this function does not return a value and the second **void** inside the brackets means it does not have any argument.

The Body of main() Function:

The body of the function **main()**is surrounded by braces (curly brackets { and }).

The left brace indicates the start of the body of the function and the matching right brace indicates the end of the body of the function.

The body of the function in the program (see Figure)consists of a single statement printf() which ends with a semicolon(;).**printf()** is the standard output function. The text to be printed is enclosed in double quotes. A statement in C is terminated with a semicolon. Therefore, a semicolon is used at the end of **printf()** statement. This program will print the message I **Love Pakistan** on the screen.

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. **(2 × 8 = 16)**

Q3. Describe the following High Level Languages.

a) C/C++ **b) Visual Basic**

c) C# **d) Java**

Ans: Popular High Level Languages (HLLs):

a) C/C++:

C language was developed in early 1970s by Dennis Ritchie at Bell Laboratories. C has become one of the most popular programming languages today. It is a highly structures programming language that is easy to understand and use. In the past, it was mainly used for writing system programs such as operating systems, compilers, assemblers, etc.

Today, it is used for writing all types of application programs as well, such as word-processing programs, spreadsheet programs, database management systems, educational programs, games, etc.

C++ was developed by Bjarne Stroustrup also at Bell Laboratories during 1983-1985. C++ is a superset

b) Visual Basic:

Visual Basic (VB) is a high level language which evolved from the earlier version called BASIC. BASIC stands for Beginner's All-purpose Symbolic Instruction Code. VB is a very popular programming language for writing Windows and Web applications. It provides a graphical development environment to programmers to develop powerful Windows and Web applications.

VB is commonly used for developing business programs such as payroll system and inventory control program. The user can also write programs related with engineering, science, arts, education, games, etc.

c) C#:

C# (pronounced as C-sharp) is a language developed in 2000 by Microsoft Corporation. It is a simple, modern, general-purpose programming language. Syntax of C# is very similar to C and C++. It also has some features of Java. It is a language that makes computer programming easy and efficient. It provides facilities to write Web applications that can be used across the Internet.

All types of programs including games, utilities, operating systems, compilers, business applications and Web based applications can be developed in C#.

d) Java:

Java is a high-level language developed by Sun Microsystems. It is very similar in syntax to C and C++. In Java, the user can write all types of programs as those written in other programming languages and small programs that can be embedded in a Web page accessed through Internet. Java is an ideal language for network computing. It is used for writing programs for a wide range of devices, computers and networks. It is widely used in Web applications. The current versions of most of the Web browsers are made Java enabled. A few browsers that support Java are Microsoft's Internet Explorer, Firefox and Mozilla.

Q4. What is C language IDE? Explain its modules in detail.

Ans: Integrated Development Environment (IDE):

Most of the new programming languages use Integrated Development Environment (IDE) to create, compile and run programs.

IDE is computer software that brings all the processes and tools required for program development into one place. IDE's aim is to make the life of programmers easier by grouping together all the tasks needed for building applications into one environment. Today's modern IDEs have user-friendly Graphical User Interface (GUI).

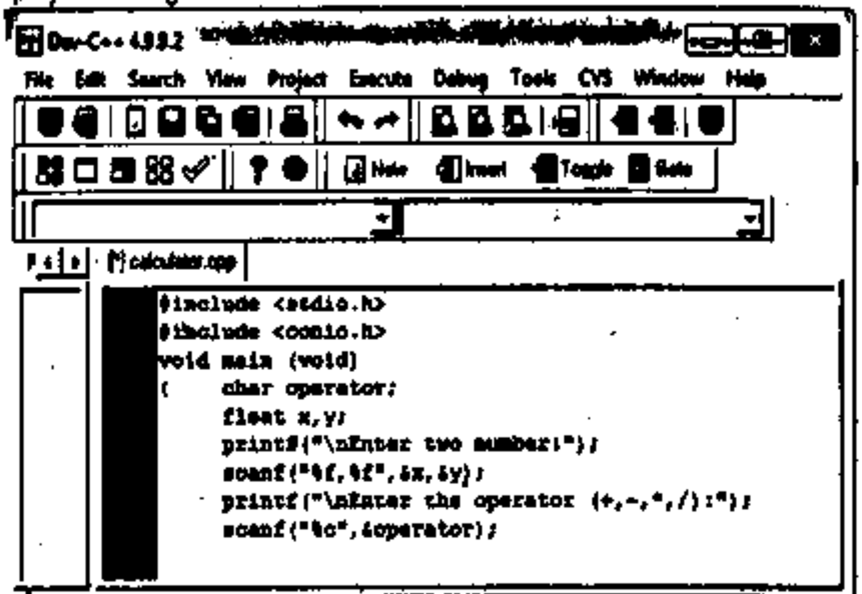
C language Integrated Development Environment (IDE) Modules:

A C language IDE consists of the following modules.

- Text Editor
- Compiler
- Linker
- Loader
- Debugger

I. Text Editor:

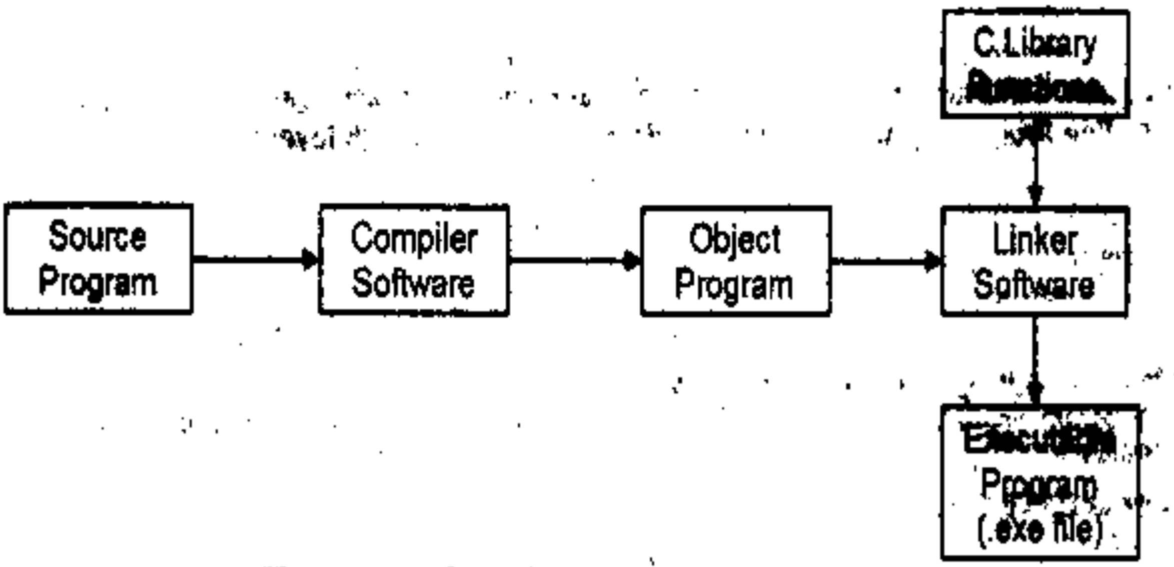
A text editor is a simple word-processor that is used to create and edit source code of a program as shown in Figure. Files created by a text editor are plain text files. Most of the editors automatically highlight compile errors to simplify removing them.


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New
Insert
Toggle
Goto
calculator.cpp

```
#include <stdio.h>
#include <conio.h>
void main (void)
{     char operator;
      float x,y;
      printf("\nEnter two number:");
      scanf("%f,%f",&x,&y);
      printf("\nEnter the operator (+,-,*,/):");
      scanf("%c",&operator);
```

ii. [illegible]

Compiler is a computer software that translates the source program (high level program) into machine code (object program) that can be understood and executed by the computer as shown in Figure. It also detects syntax errors in the source program and gives hints to the programmer to correct them. A compiler will only create object program if all the syntax errors in a program have been corrected if they existed. A program written in C language has the extension .c (for example first.c) and after compilation the object program extension will be.obj (for example first.obj)



The process of creating an executable program

iii. **Linker:**

Linker is a software that translates object program into a single executable program as shown in Figure. During this process, if it could not find the definition of a particular function that is used in the program, then it would assume that it is defined in C library. It will replace this function in the object program with the code from C library and then create a single executable program.

iv. **Loader:**

It is a software that loads programs into memory and then executes them.

v. **Debugger:**

It is a software that executes a program line by line, examines the values stored in variables and helps in finding and removing errors in programs.

Q5. What is a preprocessor directive? Explain include# preprocessor directive in detail.

Ans: Preprocessor Directives:

Preprocessor directives are instructions for the C compiler. Every C language program contains certain preprocessor directives at the beginning of the program. Before translating a C language program into machine language, the compiler of C language carries out the processor directives. These directives start with number sign (#).

The most commonly used preprocessor directives are #include and #define.

#Include Preprocessor Directive:

It has the following syntax.

#include<header file name>

When this preprocessor is carried out by the C compiler, it will search for the header file that is written within the less than (<) and greater than (>) symbols and copy it into the source file.

In the above program the header file stdio.h is used. It tells the C compiler to copy the stdio.h header file into the program. The stdio.h header file stands for standard input-output header. It includes the

IMPORTANT QUESTIONS & ANSWERS

Q1. Define computer programming.

Ans: Computer Programming:

Computer programming is the process of writing a computer program in computer language to solve a particular problem.

Computer program controls the operation of a computer and it is developed in a computer language to perform a task. Computer languages are also known as programming languages.

Q2. Define Programming Language. Explain low level and high level languages.

OR

Explain popular programming languages and their uses.

Ans: Programming Languages:

A programming language is a language which is understood by computer. It is designed to give instructions to a computer to perform a specific task. It is used to write computer programs.

Classification of Programming Languages:

Programming languages can be classified into *two categories*, that is, low level languages and high level languages.

I. Low Level Languages:

Low level language is machine-oriented language. To understand low level language, detailed knowledge of internal working of computer is required. Low level languages include *machine language* and *assembly language*.

Classification of Low Level Languages:

♦ **Machine Language:**

Programming language that is directly understood by computer hardware is known as machine language. Machine language is associated with architecture of computer. Therefore, programs written in machine language for one computer will not work on another because of design differences. It consists of zeroes and ones. It is almost impossible for humans to use machine language because it entirely consists of numbers. Therefore, practically no programming is done in machine language. Instead, assembly languages and high level languages are used.

♦ **Assembly Language:**

Assembly language consists of symbolic codes or abbreviations known as mnemonics. It was developed to make computer programming easier than machine language. The abbreviations used in assembly language make it easier to learn and write programs compared to machine language. A program written in assembly language must be converted into machine language before it is executed by computer. A program known as assembler is used to translate assembly language into machine language.

Characteristics of Assembly Language:

Some important characteristics of Assembly language are:

♦ Assembly language allows programmers to have access to all the special features of the computer they are using. Certain types of operations which are not possible in high level languages are easily programmed using assembly language.

♦ Generally a program written in assembly language will require less storage and less running time than one prepared in a high level language.

♦ Assembly languages are still the best choice in some applications but their use is gradually declining.

II. High Level Languages (HLLs):

High level languages are English-oriented languages and they are commonly used for writing computer programs. These languages use English language words such as print, goto, if, end, etc. Therefore, they are easy to learn and use.

Examples:

- A program known as compiler/interpreter is required to translate a high-level program into machine language. Coding and debugging of a high level language program is much easier than a program written in a low level language.

Classification of High Level Languages:

High-level languages can be classified into procedural, structured and object-oriented programming languages.

- **Procedural Languages:**

Procedural programming is based upon the concept of modular programming. In modular programming, programs are divided into smaller parts known as modules. Moduler programs consist of one or more modules. A module is a group of statements that can be executed more than once in a program. Each module in a program performs a specific task.

Advantages of procedural languages:

It is easy to design, modify and debug a program in a procedural language since it provides better programming facilities.

Examples of procedural languages:

Some examples of procedural languages are FORTRAN, Pascal, C and BASIC.

- **Structured Languages:**

Structured languages consist of ***three fundamental elements***, which are sequence, selection and repetition.

Sequence:

It means, writing program statements in a logical sequence. Each step in the sequence must logically progress to the next without producing any undesirable effects.

Selection:

It allows the selection of any number of statements based on the result of evaluation of a condition which may be true or false. Examples of statements that implement selection in programming are **if, else-if, switch**, etc.

Repetition (loop):

It means executing one or more statements a number of times until a condition is satisfied. Repetition is implemented in programs using statements, such as for and while loops.

Some examples of structured languages are ALGOL, PL/1, Ada and Pascal.

- **Object-Oriented Programming Languages (oops):**

Object-oriented programming (oops) refers to a programming method that is based on objects such as student, vehicle, building, etc. Object-oriented programming language provides a set of rules for defining and managing objects. An object can be considered a thing that can perform a set of activities.

For example, the object vehicle can be defined as an object that has number of wheels, number of doors, color, number of seats, etc. The set of activities that can be performed on this object include Steer, Accelerate, Brake, etc.

Complicated and large computer programs are difficult to design, develop, maintain and debug. The concept of object-oriented programming solves this problem.

The most widely used object-oriented programming languages are C++, C#, php and Java.

Q3. What is programming environment?

OR

Describe the concept of integrated development environment.

Ans: Programming Environment:

Programming environment is the set of processes and programming tools used to develop computer programs. In the past, programmers used various standalone programming tools for developing computer programs. These included editor, compiler, linker, debugger, etc. Using separate programs provided a difficult and time consuming environment for creating computer programs.

Q4. Explain the different data types used in C programs.

Ans: Data Types Used In C Programs:

C provides three main data types for variables, that is, integer, floating-point and character variables.

- **Integer Data Type:** Integer variable declaration statement has the form:

Type specifier Variable;

For example: Int sum;

The declaration consists of the type name, int, followed by the name of the variable, **sum**. All the variables used in a C program must be declared. If there are more than one variable of the same type, separate the variable names with commas as shown in the following declaration statement.

Int a,b, sum, product;

Declaring a variable tells the computer the name of the variable and its type. This enables the compiler to recognize the valid variable names in program. This is very helpful if the user types the wrong spellings of a variable name in his program, the compiler will give an error message indicating that the variable is not declared.

Declaration of an integer variable can begin with the type qualifiers, **short**, **long, unsigned or signed.**

Some examples are: **short Int num;**
long Int sum, avg;
unsigned Int count;
short unsigned Int count;

Type qualifiers refer to the number of bytes used for storing the integer on a particular computer. Refer to computer manual to find out the sizes of the computer being used. Generally, the number of bytes set aside by the compiler for the above type qualifiers are as given in table.

Integer data types used in C:

Variable Declaration	No. of Bytes	Range
--	2	-32,768 ~ +32,767
long int	4	-2,147,483,648 ~ +2,147,483,647
unsigned int	2	0 ~ 65,535

Here, **unsigned** implies that the value of variable is greater than or equal to zero. The qualifier, **signed**, is rarely used since by default, an **Int** declaration assumes a **signed** number. Note that in the above examples, the word **int** may be omitted when it begins with**long** or **unsigned** type qualifiers in the declaration statements.

For example you can write: **short num;**

- **Floating-point Data Type:**

Floating-point variables are used for storing floating-point numbers. Floating point numbers are stored in memory in two parts. The first part is the **mantissa** and the second part is the **exponent.**

The **mantissa** is the value of the number and the **exponent** is the power to which it is raised. Some examples of floating-point variable declaration statements are:

float base,height;

double float a,b,total;

long double float size,x,y;

Usually the number of bytes set aside by the compiler for the above floating-point type qualifiers are as given in table.

Floating-point data types used in C:

Variable Declaration	No. of Bytes	Range
float	4	$10^{-38} \sim 10^{38}$ (with 6 or 7 digits of precision)
double float	8	$10^{-308} \sim 10^{308}$ (with 15 digits of precision)

Here, precision means the number of digits after the decimal point. The word **float** may be omitted when it is preceded by **double or long double** type qualifiers in the declaration statements.

There is another method of representing floating-point numbers known as exponential notation.

Example:

For example, the number 23,688 would be represented as 2.3688e4. Here, 2.3688 is the value of the number (mantissa) and 4 is the exponent(e). The exponent can also be negative.

Example:

For example, the number 0.0005672 is represented as 5.672e-4 in exponential notation. Exponential notation is used to represent very large and very small numbers. In C, a floating point number of type **float** is stored in four bytes. Three bytes for the **mantissa** and one byte for the **exponent** and provides 6 or 7 digits of precision.

- **Character Data Type:**

Character variables are used to store character constants. Examples of declaration of character variables are:

char ch,letter,a;

A character variable can only store one character. The compiler sets aside one byte of memory for storing a character in a character variable.

Q5. List the uses of C language.

Ans: Uses of C language:

C is a popular and widely used programming language for creating computer programs. A large variety of application programs and many modern operating systems are written in C.

Q6. Explain C language character set.

Ans: C Language Character Set:

The C Language character set includes:

Letters:

C language comprises the following set of letters to form a standard program. They are:

- A to Z in Capital letters.
- a to z in Small letters.
- In C programming, small latter and caps latter are distinct.

Digits:

- C language comprises the following sequence of numbers to associate the letters. 0 to 9 digits.

Special Characters:

C language contains the following special character in association with the letters and digits.

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
~	Tilde	_	Underscore	]	Right bracket
!	Exclamation mark	+	Plus sign	:	Colon
#	Number sign	|	Vertical bar	"	Quotation mark
$	Dollar sign	\	Backslash	;	Semicolon
%	Percent sign	'	Apostrophe	<	Opening angle bracket
^	Caret	-	Minus sign	>	Closing angle bracket
&	Ampersand	=	Equal to sign	?	Question mark
*	Asterisk	{	Left brace	,	Comma
(	Left parenthesis	)	Right parenthesis	.	Period

Q7. Define header files.

Ans: Header Files:

C language contains a number of standard functions in library file that perform various tasks in C programs. These tasks include all the input/output operations and all the math operations. Library file contains header files and each header file contains a set of functions.

Q8. What is the structure of a C program?

Ans: Structure of a C Program:

The format according to which a program is written is called the structure of program. The following is the structure of a C program.

```
Preprocessor directives
Global declarations
Main function
{
      Local declarations  }
      Statements          } Body of main() function
}
User-defined function
Function 1    }
Function 2    } (option to user)
Function 3    }
```

Q9. Explain the types of comments in C language.

Ans: Types of Comments:

There are two types of comments.

i. Single line comment
ii. Multiple line comment

i. Single line comment:

The // is used as single line comment. The syntax of single line comment is:

// comment

An example is:

// Programmer: M. Sajjad Heder

ii. Multiple line comment:

The /* .... */ is used for multiple line comments.

Comment can span to multiple lines as shown in the following example.

```
/* Programmer: M. Sajjad Heder
Programming Language: C
Date Program was Written: 7-01-2019 */
```

GUESS PAPER & MODEL PAPER # 3

Based on Chapter # 03 (Reduced Syllabus)

INPUT AND OUTPUT HANDLING

SECTION-A (Marks 12)

Time allowed: 20 Minutes **Marks: 12**

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. Which function is used for output purpose in C language?
A. printf() B. scanf() C. input() D. getch()

ii. What will be the output of the expression, 5+3*3-1?
A. 16 B. 13 C. 23 D. 12

iii. Which character terminates a C statement?
A. Colon B. Semicolon C. Period D. Comma

iv. Which format specifier is used to print or read a floating-point value in decimal notation?
A. %d B. %g C. %f D. %e

v. Which escape sequence is used to move cursor to the beginning of current line?
A. \a B. \r C. \n D. \b

vi. Which of the following is an arithmetic operator?
A. % B. <= C. && D. +=

vii. Which of the following is a logical operator?
A. % B. <= C. && D. +=

viii. Which statement is equivalent to "k = k + a;"?
A. k+=a; B. k=+a; C. k++a; D. k=a++;

ix. Which of the following is an increment operator?
A. + B. += C. ++ D. =+

x. Which of the following operator has the highest precedence?
A. && B. <= C. = D. *

xi. The __________ function is used to read a character string from the keyboard.
A. scanf() B. gets() C. getch() D. getche()

xii. Which function is used to print text and values on the screen in a specified format?
A. printf() B. putchar() C. puts() D. getche()

COMPUTER SCIENCE SSC-II

Time allowed: 2:40 Hours **Total Marks Sections B and C: 43**

NOTE: Answer any nine parts from Section 'B' and any two questions from Section 'C' on the separately provided answer book. Use supplementary answer sheet i.e. Sheet-B if required. Write your answers neatly and legibly.

SECTION – B (Marks 27)

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Why format specifier is used? Explain with examples.

ii. Why escape sequence is used? Explain with examples.

iii. What is the purpose of printf() function? Explain with an example.

vi. What will be the output of the following program?

```
#include <stdio.h>
void main(void)
{
        int x,y,z1,z2,z3,z4;
        x=17;
        y=5;
        z1=x/y;
        printf("\nz1=%d",z1);
        z2=x%y;
        printf("\nz2=%d",z2);
        z3=++x;
        printf("\nz3=%d",z3);
        z4=y++;
        printf("\nz4=%d",z4);
}
```

vii. What will be the output of the following program?

```
# include <stdio.h>
void main(void)
{
        int b;
        floata,c,d,e,f;
        a=14.84;
        b=7;
        c=a-b;
        printf("\nc=%f",c);
        d=a/b;
        printf("\nd=%f",d);
        e=a-b*3;
        printf("\ne=%f",e);
        f=(a+b)/2;
        printf("\nf=%f",f);}
```

viii. Write a program that reads the length and width of a rectangle and prints its area.

ix. Write a program that reads the length of one side of a cube and prints its voleme.

x. Write a program that reads temperature in Celsius, converts it into Fahrenheit and prints on the screen.

xi. What is the statement terminator in C?

OR

Write down the uses of statement terminator in C.

xii. What are operators of C language? List the types of operators.

xiii. Differentiate between assignment operator and equal to operator?

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. Describe how basic and compound assignment operators are used?

Q4. Describe the functions of the following operators?

i) Relational operators ii) Logical operators iii) Conditional operator

Q5. Write a program that reads three numbers and prints their sum, product and

SOLUTION OF GUESS PAPER & MODEL PAPER # 3 (Reduced Syllabus)

SECTION- A (MCQs)

i. B	ii. B	iii. B	iv. C	v. B	vi. A
vii. C	viii. A	ix. C	x. D	xi. B	xii. A

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Why format specifier is used? Explain with examples.

Ans: Format Specifiers:

A format specifier is computer code that tells about the data type, field width and the format according to which a value is to be printed or read from an input device. A list of commonly used format specifiers is given below.

%d	decimal integer
%i	integer
%ld	long decimal integer
%f	floating-point (decimal notation)
%e	floating-point (exponential notation)
%c	single character

ii. Why escape sequence is used? Explain with examples.

Ans: Escape Sequence:

The special characters used in C language to control printing on the output device are called escape sequences. These characters are not printed. These are used inside the control string.

An escape sequence is a combination of a backslash (\) and a code character. The backslash is called the control character. A list of commonly used escape sequences is given in table with their meanings.

Escape sequences used in C language:

Escape Sequence	Meaning
\a	Produces alert (bell) sound
\b	Moves cursor backward by one position
\n	Moves cursor to the beginning of next line
\r	Moves cursor to the beginning of current line
\t	Moves cursor to the next horizontal tabular position
\\	Produces a backslash
\'	Produces a single quote
\"	Produces a double quote

iii. What is the purpose of printf() function? Explain with an example.

Ans: Purpose of printf() Function:

The printf() function is used to print text and values on the screen in a specified format.

Syntax: The general syntax of this function is:

printf(control_string, list of arguments);

Example: Following is an example of **printf()** function.

```
printf("\nThe value of a is %d and the value of b is %d.",a ,b);
```

iv. Differentiate between printf() and scanf() function

v. Evaluate the following expressions.

a) 7+5*(3+4) b) 100/10/4

c) 50%13%3 d) 30/7*3-6

Ans:

a) 7+5*(3+4)
= 7+5*7
= 7+35
= 42

b) 100/10/4
= 10/4
= 2.5

c) 50%13%3
= 11%3
= 2

d) 30/7*3-6
= 4.28*3-6
= 12.8-6
= 6.8

vi. What will be the output of the following program?

```
# include <stdio.h>
 void main(void)
{
        int x,y,z1,z2,z3,z4;
        x=17;
        y=5;
        z1=x/y;
        printf("\nz1=%d",z1);
        z2=x%y;
        printf("\nz2=%d",z2);
        z3=++x;
        printf("\nz3=%d",z3);
        z4=y++;
        printf("\nz4=%d",z4);
}
```

Ans: Output:

```
Z1 = 3
Z2 = 2
Z3 = 18
Z4 = 5
```

vii. What will be the output of the following program?

```
# include <stdio.h>
void main(void)
{
        int b;
        floata,c,d,e,f;
        a=14.84;
        b=7;
        c=a-b;
        printf("\nc=%f",c);
        d=a/b;
        printf("\nd=%f",d);
        e=a-b*3;
        printf("\ne=%f",e);
        f=(a+b)/2;
```

```
d = 2.120000
e = 6.160000
f = 10.920000
```

```
    printf("Enter length of rectangle: ");
    scanf("%f",&length);
    printf("Enter width of rectangle: ");
    scanf("%f",&width);
    area= length * width;
    printf("Area of rectangle = %f sq. units ", area);
  }
```

ix. Write a program that reads the length of one side of a cube and prints its volume.

Ans:

```
#include <stdio.h>
#include <conio.h>
main()
{
        float side, volume;
        printf("Enter length of any side of cube\n");
        scanf("%f", &side);
        volume = side*side*side;
        printf("Volume of Cube : %0.4f\n", volume);
 }
```

x. Write a program that reads temperature in Celsius, converts it into Fahrenheit and prints on the screen.

Ans:

```
#include<stdio.h>
#include<conio.h>
main()
{
        float fahren, celsius;
        printf("Enter the temperature in celsius\n");
        scanf("%f", &celsius);
        fahren =(9.0/5.0) * celsius + 32;
        printf("%.2fC is equal to %.2fF\n", celsius, fahren);
}
```

xi. What is the statement terminator in C?

OR

Write down the uses of statement terminator in C.

Ans: Statement Terminator:

Semicolon (;) is entered at the end of a statement in C language. It indicates the compiler that the statement ends here. If a statement is not terminated with semicolon, C compiler will give an error message during compilation and the program will not be compiled.

xii. What are operators of C language? List the types of operators.

Ans: Operators of C Language:

Expressions consist of constants and variables combined together with operators. An operator is a

The following types of operators are commonly used in C language.

- Arithmetic operators
- Assignment operators
- Relational operators
- Logical operators
- Increment and decrement operators

xiii. Differentiate between assignment operator and equal to operator?

Ans: Difference between Assignment Operator and Equal to Operator:

The assignment operator (=) is used to assign a value to a variable whereas the equal to operator (==) is used to compare two values of same data type.

For example:

```
a=5;
c=b;
z=x+y;
```

In the above statements, variable **a** is assigned the value 5, **c** is assigned the value stored in variable b and **z** is assigned the sum of values stored in variable **x** and **y**.

The relational operator (= =) is used to build a condition based on which computer takes some action.

For example:

```
a= =1
c= = a+b
```

In the first condition, if the value of **a** is equal to 1 then the condition is true otherwise it is false. In the second condition, the equal to operator is used to check whether the value of **c** is equal to the sum of **a** and **b**. If it is equal then the condition is true otherwise, it is false.

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. (2 x 8 = 16)

Q3. Describe how basic and compound assignment operators are used?

Ans: Assignment Operators:

Assignment operators are used to assign values to variables used in computer programs. C language provides three types of assignment operators. These are basic assignment operator, compound assignment operators and increment/decrement operators

Basic Assignment Operator:

The basic assignment operator is =. This is used to assign value of an expression to a variable. It has the general form: **variable = expression**

where expression may be a constant, another variable to which a value has previously been assigned or a formula to be evaluated.

For example: **sum = a + b;**

Compound Assignment Operators:

In addition to =, there are a number of assignment operators unique to C. These include +=, - =, *=, /= and %=. Suppose **op** represents an arithmetic operator. Then, the compound assignment operator has the following general form to assign value of an expression to a variable. **variable op = expression**

This is equivalent to: **variable = variable op expression**

For example, consider the following statement: **sum = sum + n;**

This assignment statement could be written using a compound assignment operator as: **sum += n;**

The effect is exactly the same but the expression is more compact. Some more examples are:

sum - = n	is equivalent to	**sum = sum – n**
prod *= n	is equivalent to	**prod = prod * n**

Q4. Describe the functions of the following operators?

i) Relational operators ii) Logical operators iii) Conditional operator

Ans: i) Relational operators:

Relational operators are used to compare two values of the same type. These are used in expressions when a decision is to be based on a condition. After evaluation of a relational expression, the result produced is either True or False. Relational operators are used in programming for decision making.

Types of Relational Operators:

Six types of relational operators are available in C language. These are described in table.

Operator	Definition
= =	equal to
!=	not equal to
<	less than
>	greater than
<=	lessthan or equal to
>=	greater than or equal to

The following are some examples of relational operators.

```
c>=a+b
x<5.3
n==20
count!=10
```

If a has the value 25, b has the value 10 and c has the value 28 then first expression is false since 28 is not greater than or equal to 35.

In the second expression, if x has the value 4.5, the expression is true because x is less than 5.3.

In the third expression, if n is equal to 20 then the expression will be true. For any value other thaṅ 20, the expression will be false.

In the last expression if count is any number other than 10 then the expression will be true. It will only be false when count is equal to 10.

ii) Logical operators:

Logical operators are used for building compound conditions. We have seen before that a single condition is built using a relational operator in an expression. If we need to build more than one condition for some action to take place in programming, then we have to form compound condition.

Types of Logical Operators:

There are three types of logical operators. These are described in table.

Operator	Definition
&&	AND
||	OR
!	NOT

Logical AND (&&) Operator:

It is used to form compound condition in which two relational expressions are evaluated. One relational expression is to the left and the other to the right of the operator. If both of the relational expressions (conditions) are true then the compound condition is considered true otherwise it is false.

Syntax: *Expression1 && Expression2*

Truth table for AND operator is shown here under:

Expression-1	Expression-2	Expression-1 && Expression-2
True	True	True
True	False	False
False	True	False

Example:

Consider the following compound condition:

(a>=1) && (a<=10)

(a>b) && (c==10)

The following compound condition will check whether the character stored in variable ch is a lowercase letter or not.

(ch>='a') && (ch<='z')

Logical OR (||) Operator:

Logical OR operator is also used to form a compound condition. Just like the logical AND operator, one relational expression is to the left and the other to the right of the OR operator. The compound condition is true if either of the conditions is true or both conditions are true. It is considered false only if both of the conditions are false.

Syntax: ***Expression1 || Expression2***

Truth table for AND operator is shown here under:

Expression-1	Expression-2	Expression-1 || Expression-2
True	True	True
True	False	True
False	True	True
False	False	False

Example:

(n<10)||(n>25)

Suppose, the value of **n** is 5, then the expression will be considered true because one of the two conditions is true. If the value of n is 28 then also the compound condition will be true. If the value of **n** is 12 then the expression will be false since both conditions are false.

The next compound condition will be true if **a** is greater than **b** or **c** is equal to 10. It will also be true if both conditions are true, that is, **a** is greater than **b** and c is equal to 10. It will only be false if **a** is not greater than **b** and at the same time **c** is not equal to 10.

(a>b) || (c= =10)

Logical OR condition is used when we wish to perform an operations if one of the two conditions is true or both of the conditions are true.

Logical NOT (!) Operator

The logical NOT operator is used with a single expression (condition) and evaluates to true if the expression is false and evaluates to false if the expression is true. In other words, it reverses the result of a single expression.

Syntax: *!Expression1*

Truth table for AND operator is shown here under:

Expression	!Expression
True	False
False	True

For example, the expression:

!(a<b)

will be true if **a** is not less than b. In other words, the condition will be true if **a** is greater than or equal to b. The same condition can also be written as given below which is easy to understand.

(a>=b)

When this statement is executed, the condition is evaluated. If it is true, the entire conditional expression takes on the value of expression1. If it is false, the conditional expression takes on the value of expression2. The entire conditional expression takes on a value and can therefore be used in an assignment statement. Consider the following example.

```
a = (k>15)? x*y : x+y;
```

This statement will assign the product of x and y to the variable a, if k is greater than 15, otherwise it will be assigned the sum of x and y. This expression is equivalent to the following if-else statement which will be explained in the next unit.

```
If (k>15)
      a = x*y;
else
      a = x+y;
```

Some programmers may prefer to use the above if-else statement rather than using the conditional operator because it is easy to understand.

Q5. Write a program that reads three numbers and prints their sum, product and average.

Ans:

```
#include <stdio.h>
int main()
{
      int num1, num2, num3, sum, prod, avg;
      printf("Enter the three numbers separated by ','\n");
      scanf("%d, %d, %d", &num1, &num2, &num3);
      sum = num1 + num2 + num3;
      prod=num1 * num2 * num3;
      avg=sum/3;
      printf("the sum of the three numbers is : %d\n", sum);
      printf("the prod of the three numbers is : %d\n", prod);
      printf("the average of the three numbers is : %d\n", avg);
}
```

IMPORTANT QUESTIONS & ANSWERS

Q1. Define input functions.

Ans: Input Functions:

In computer programming input means to feed data into a program through an input device. C language provides different functions to input data. The most common is the scanf() function.

Q2. Define output functions.

Ans: Output Functions:

In computer programming, output means to display information on screen or print on printer. C language provides different functions to output information on computer screen. The most common is the printf() function.

Q3. Briefly explain the printf() function with the help of an example.

Ans: The printf() Function:

The printf() function is used to print text and values on the screen in a specified format.

Syntax:

The general syntax of this function is:

```
printf(control_string, list of arguments);
```

The **control_string** consists of text, the format specifiers and the escape sequence. It is written within

List of arguments consists of a list of variable- or arithmet-c expressions, separated by commas, whose values are to be printed. The values are printed according to the corresponding format specifier. The first format specifier applies to the first argument, the second to the second argument and so on. The arguments in

```
printf("\nThe value of a is %d and the value of b is%d ",a,b);
```

Assuming that a has the value of 3 and b has the value of 4, then the following message will be printed by the above statement on a new line.

The value of a is 3 and the value of b is4.

In the above print statement, \n is an escape sequence and **%d** is a format specifier. The escape sequence \nis used to start printing on a new line. The format specifier**%d** is repla:ed by the values stored in variables **a** and b in the order in which they are written. The format specifier**%d** is used for printing values stored in integer variables.

Q4. Briefly explain the scanf() function with the help of an example.

Ans: The scanf() Function:

The scanf() function is used to get values into variables from the keyboard during the execution of a program. The value is input into a variable in a specified format.

Its **syntax** is: **scanf("formate specifiers", list of variables**):

The **formate specifiers** specifies the format of the variables. These are are written within double quotes with or without spaces.

The **'list of variables'** consists of a list of variables. separated by commas, into which the values are to be entered. In C language, ampersand **(&)** symbol is used in **scanf**() function before the name of the variable to which a value is to be assigned. Ampersand (&) sign refers to the memory location where the variable is going to store.

For example, to input values into two integer variables **a** and **b**, the **scanf()** function is written as:

scanf("%d,%d", &a, &b):

Here the format specifier %d is used twice in control string for the variables **a** and **b**. When this statement is executed the values may be entered by separating them by comma and the user has to press the Enter key after entering the data.

Q5. Define arithmetic operators and its types. Also explain how to use them in a program.

Ans: Arithmetic Operators:

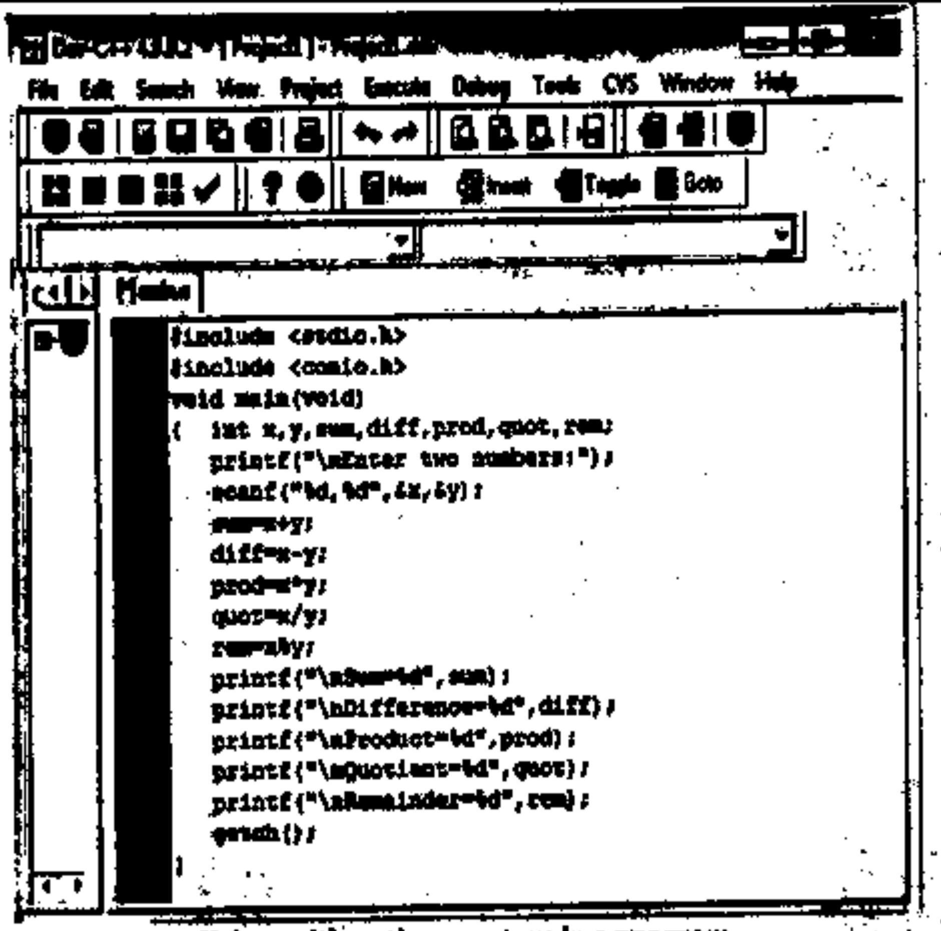
Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and also to find the remainder obtained when an integer is divided by another integer.

Types of Arithmetic Operators:

The types of arithmetic operators used in C programming are described with their operations in table.

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (Modulus) Operator

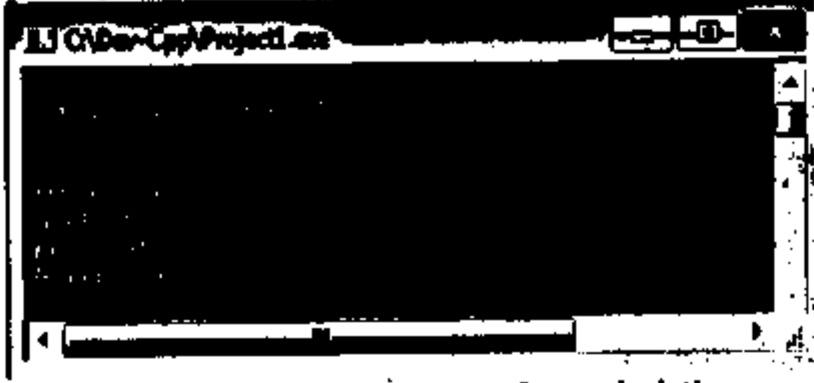
The program in Fi


File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto

```
#include <stdio.h>
#include <conio.h>
void main(void)
{   int x,y,sum,diff,prod,quot,rem;
    printf("\nEnter two numbers:");
    scanf("%d, %d",&x,&y);
    sum=x+y;
    diff=x-y;
    prod=x*y;
    quot=x/y;
    rem=x%y;
    printf("\nSum=%d",sum);
    printf("\nDifference=%d",diff);
    printf("\nProduct=%d",prod);
    printf("\nQuotient=%d",quot);
    printf("\nRemainder=%d",rem);
    getch();
}
```

Using arithmetic operators in a program

In this program when **x/y** is performed it will give an integer result because the fractional part is truncated when the two operands are of type integer. Moreover, the remainder operator will give the remainder after dividing **x** by **y** when **x%y** is performed. The output of the program is shown in Fig.



Use of arithmetic operators to perform calculations

Q6. Explain the use of increment and decrement operators in C with the help of examples.

Ans: Increment and Decrement Operators:

Increment operator is ++ and decrement operator is - -. These are defined in table.

Operator	Definition
++	Increment by 1
- -	Decrement by 1

Increment and decrement operators

Examples:

++n and n++ are both equivalent to n = n + 1 (or n+=1)

- - n and n- - are both equivalent to n = n – 1 (or n-=1)

When increment or decrement operator is written before the variable, it is known as **prefix** and when it

The statement: a = ++n; will first increment n and then assigns the value 4 to a.
But the statement: a = n++; will first assign the value 3 to a and then increments n to 4. In both cases n has the value 4. The same rule applies to - -n and n- - as well.

Q7. Differentiate between Unary and Binary Operators.

Ans: Difference between Unary and Binary Operators:

The operators that work with a single operand are known as unary operators whereas operators that work with two operands are known as binary operators. Unary operators are -, ++, -- and the logical operator! (NOT). Binary operators are -, +, *, \, % and logical operators && (AND) and || (OR).

Some **examples of unary operators** are:

```
a= -b;
k++;
- -x;
```

Some **examples of binary operators** are:

```
a=b+c;
z=x*y;
k=d%e;
```

Q8. Define and explain the order of precedence of operators.

Ans: Order of Precedence of Operators:

Order of precedence of operators is the rule that specifies the order in which operations are to be performed in an expression. The order of precedence is similar to that used in algebraic formulas.

The order of precedence of operators is shown in table. The operator that has the highest precedence is written at the top and the one with the lowest precedence is written at the bottom.

Precidence	Operator	Description
1.	++, --	Increment and Decrement (Prefix or Postfix)
2.	*, /, %	Multiplication, Division and Remainder
3.	+, -	Addition and Subtraction
4.	<, <=, >, >=	Relational Operators
5.	= =, !=	Equal to and Not Equal to
6.	!	Logical NOT
7.	&&	Logical AND
8.	||	Logical OR
9.	=, *=, /=, %=, +=, -=	Assignment Operators

Order of precedence of operators

Q9. Why Integer Format Specifiers is used?

Ans: Integer Format Specifiers (%d, %ld and %l):

The format specifier %d is used to read or print a decimal integer and the format specifier% ld is used with long integers.

Q10. Why Floating-point Format Specifiers is used? Explain with examples.

Ans: Floating-point Format Specifiers (%f, %e)

The format specifier %f is used to read and print floating-point numbers in decimal notation with a precision of 6 digits after the decimal point.

The format specifier %e is used to read and print floating-point numbers in exponential notation.

GUESS PAPER & MODEL PAPER # 4

Based on Chapter # 04 (Reduced Syllabus)

CONDITIONAL CONTROL STRUCTURE

SECTION-A (Marks 12)

Time allowed: 20 Minutes **Marks: 12**

NOTE: Section-A is compulsory. All parts of this section are to be answered on the question paper itself. It should be completed in the first 20 minutes and handed over to the Centre Superintendent. Deleting/overwriting is not allowed. Do not use lead pencil.

Q1. Circle the correct option i.e. A / B / C / D. Each part carries one mark.

i. For which purpose if structure is used in programming?

A. Repetition B. Selection
C. Sequence D. Input of data

ii. Which statement is suitable to use in a situation where there are only two choices based on a condition?

A. if statement B. if-else-if statement
C. if-else statement D. switch statement

iii. Which statement can be used in place of switch statement?

A. if statement B. if-else statement
C. if-else-if statement D. conditional operator

iv. Which statement can be used in place of conditional operator?

A. if statement B. if-else statement
C. else-if statement D. switch statement

v. Which statement is used to exit from the body of switch statement?

A. default B. continue C. exit D. break

vi. Which of the following is a multiple selection statement?

A. if statement B. if-else statement
C. if-else-if statement D. none of these

vii. Which of the selection structures tests only for equality?

A. if statement B. if-else statement
C. else-if statement D. switch statement

viii. What will be printed when the following code is executed?

```
x=1;
switch(x)
{       case 1:
        case 2:
        case 3:
                printf("\n x is a positive number");
                break;
        default
                printf("\n value of x is 1");          }
```

A. value of x is 1 B. x is a positive number
C. Nothing will be printed D. It will give error

ix. Statement which is used to make choice between two options and only option is to be

x. Which of the following is an invalid if-else statement?
   A. if (if (a ==1)){} B. if (func1 (a)){}
   C. if (a){} D. if ((char) a){}

xi.
```
#include <stdio.h>
int i;
int main()
{
  if (i);
  else
    printf("Else");
  return 0;
}
```
What is correct about the above program?
   A. if block is executed. B. else block is executed.
   C. It is unpredictable as i is not initialized. D. Error: misplaced else

xii. The switch statement is similar to:
   A. if statement B. if-else statement
   C. else-if statement D. switch statement

COMPUTER SCIENCE SSC-II

Time allowed: 2:40 Hours Total Marks Sections B and C: 43

NOTE: Answer any nine parts from Section 'B' and any two questions From Section 'C' on the saperately provided answer book. Use supplementary answer sheet i.e. Sheet-B if required. Write your answers neatly and legibly.

SECTION – B (Marks 27)

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 × 3 = 27)

i. Differentiate between if and if-else selection structures.

ii. Differentiate between else-if and switch selection structures.

iii. What is nested selection structure?

iv. Write the following statement using if-else statement.

v. Write the following statement using conditional operator.
```
If (x>y)
        z=(x+y)/3;
else
        z=x-5*y;
```

vi. What will be the output of the following code?
```
int n, count, sum;
n=28; count=15; sum=30;
if (n<25)
{       count=count+5;
        printf("\nCount=%d",count);    }
else
{       count=count-5;
        sum=sum+n;
        printf("\nCount=%d",count);
        printf("\nSum=%d",sum);    }
```

vii. What will be the output of the following code?
```
charch;
```

```
        case 'b':
                printf"\nHave a Nice Day!  "; break;
        case 'c':
        case 'd':
        case 'e':
                printf("\n  Good Bye!  "); break;
    }
```

viii. What is the advantage and limitation of switch statement?

ix. Explain the Use of conditional operator in program.

x. Define a control statement.

xi. Define a conditional statement.

xii. What is the purpose and structure of if statement? Explain with the help of examples.

xiii. What is the purpose and structure of if-else statement? Explain with the help of examples.

SECTION – C (Marks 16)

Note: Attempt any TWO questions. All questions carry equal marks. (2 x 8 = 16)

Q3. What is control structure? Explain conditional control structure with examples.

Q4. What is the purpose of switch () statement? Explain with the help of one example.

Q5. What is the purpose and structure of if-else-if statement? Explain with the help of examples.

SOLUTION OF GUESS PAPER & MODEL PAPER # 4 (Reduced Syllabus)

SECTION- A (MCQs)

i. B	ii. C	iii. C	iv. B	v. D	vi. C
vii. D	viii. B	ix. A	x. A	xi. B	xii. C

SECTION- B

Q2. Answer any NINE parts. The answer to each part should not exceed 3 to 4 lines. (9 x 3 = 27)

i. Differentiate between if and if-else selection structures.

Ans: **If Selection Structure:**

The if statement has the following **general form / Syntax.**

```
If (condition)
{
        Block of statements
}
```

If-else Selection Structure:

The **if-else** statement is used in situation where some code is to be executed if a condition is true and, some other code is to be executed if the condition is false.

The **if-else** statement has the following **general form / Syntax.**

```
If (condition)
{
        Block of statements
}
else
{
```

ii. Differentiate between else-if and switch selection structures.

Ans: else-if Selection Structures:

The **if-else** statement is used in situation where some code is to be executed if a condition is true and some other code is to be executed if the condition is false.

The **if-else** statement has the following **general form / Syntax.**

```
if (condition)
{
        Block of statements
}
else
{
        Block of statements
}
```

- When **if-else** statement is executed, the condition is evaluated.

Switch Selection Structure:

The **switch** statement has the following general form.

```
switch (expression)
{
case const-1:
        statements;
        break;
case const-2:
        statements;
        break;
        .
        .
        .
default:
        statements;
}
```

- The **switch** statement is similar to the **else-if** statement. It is used when multiple choices are given and one choice is to be selected.

iii. What is nested selection structure?

Ans: Nested Selection Structure:

The **selection** structure that is within another selection structure is known as nested selection structure. Sometimes, in computer programming, it is required to use a selection structure within another selection structure. This is also supported in C language. In C language, the programmer can have a selection structure (**if, if-else, else-if** or **switch statement**) within another selection structure.

iv. Write the following statement using if-else statement.

```
k = (a+b>20)? a+3*b: a-b;=
```

Ans:

```
If (a+b)>20
        K=a+3*b;
Else
        K=a-b;
```

v. Write the following statement using conditional operator.

```
if (x>y)
        z=(x+y)/3;
else
```

vi. What will be the output of the following code?

```
int n, count, sum;
n=28; count=15; sum=30;
if (n<25)
{       count=count+5;
        printf("\nCount=%d",count);   }
else
{       count=count-5;
        sum=sum+n;
        printf("\nCount=%d",count);
        printf("\nSum=%d",sum);   }
```

Ans: Count = 10
Sum = 58

vii. What will be the output of the following code?

```
charch;
ch='c';
switch(ch)
{       case 'a':
                printf("\nGood Morning!  "); break;
        case 'b':
                printf"\nHave a Nice Day!  "; break;
        case 'c':
        case 'd':
        case 'e':
                printf("\n  Good Bye!  "); break;
}
```

Ans: Good Bye!

viii. What is the advantage and limitation of switch statement?

Ans: Advantage and Limitation of Switch Statement:

- The switch statement allows a variable to be compared against a list of constant values. When there is a match to a case, the statements following that case will execute until a break statement is reached. This makes the logic of program simple and easy to understand.
- The switch statement has a limitation. It is not allowed to use relational operators in the expression of switch statement.

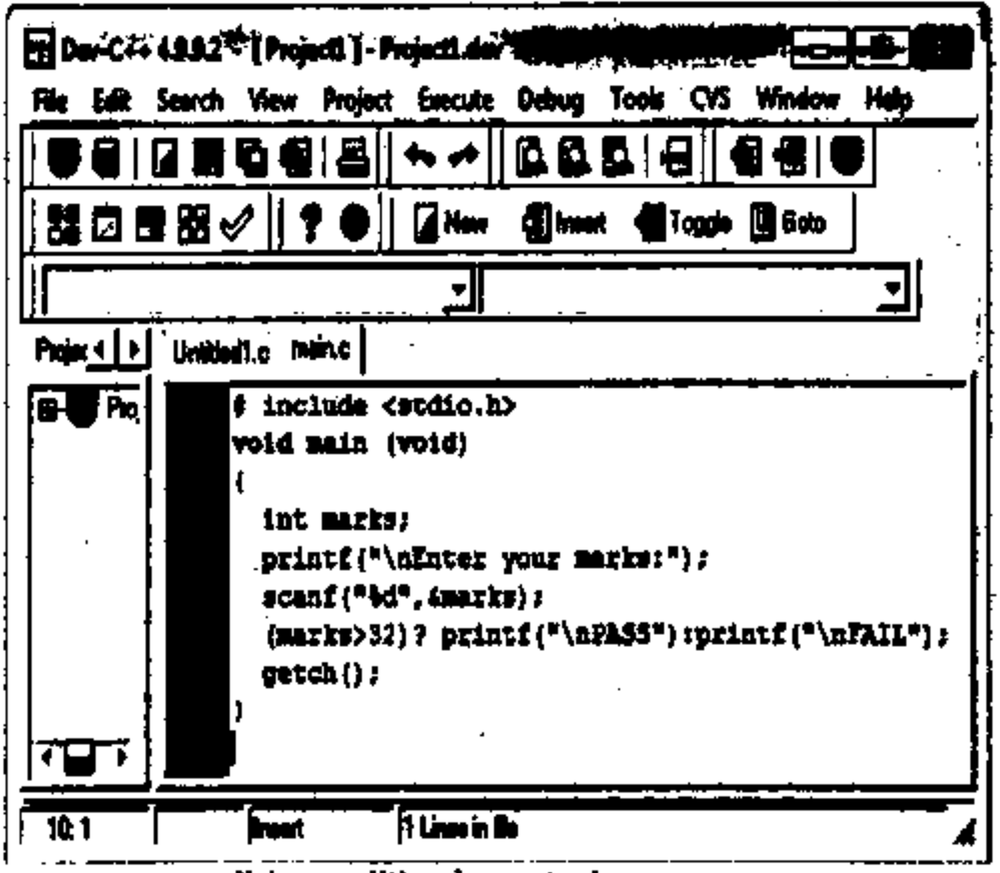
For example, to check for passing marks, the condition, (marks>32) cannot be used in switch statement. It is also not possible to check for a range such as ((marks>=70)&&(marks<=80)) in a switch statement, for this, the programmer has to use if, if-else or else-if statement.

ix. Explain the Use of conditional operator in program.

Ans: Using Conditional Operator in Program:

Program:

The program in Figure., reads marks and prints the message "PASS" or "FAIL", using conditional operator instead of if-else statement.

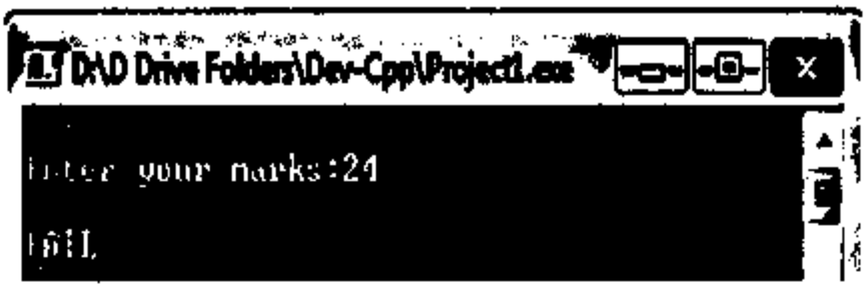

Dev-C++ 4.9.9.2 [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Projec
Untitled1.c main.c
10: 1
Insert
11 Lines in file

```
# include <stdio.h>
void main (void)
{
  int marks;
  printf("\nEnter your marks:");
  scanf("%d",&marks);
  (marks>32)? printf("\nPASS"):printf("\nFAIL");
  getch();
}
```

Using conditional operator in a program

- When this program is executed, it will ask the user to enter the marks and store it in the variable **marks.**
- The condition **(marks>32)** will be evaluated.
- If the condition is true, that is, marks are greater than 32 then the first print statement will be executed and the message "PASS" will be printed.
- If the condition is false then the second print statement will be executed and the message "FAIL" will be printed.

Execution of the program is shown in Fig.



```
Enter your marks:24

FAIL
```

Execution of Program

x. Define a control statement.

Ans: Control Statement:

A control statement is an instruction which determines the sequence of execution of other statements. In other words, it controls the flow of execution of program statements.

xi. Define a conditional statement.

Ans: Conditional Statement:

A conditional statement is an instruction in a programming language that contains a condition. When a conditional statement is executed, first the condition is evaluated and then based on the result (true or false), a particular statement or a set of statements is executed. Conditional statements of C language are ***if, if-else, else-if*** and **switch** statements.

xii. What is the purpose and structure of if statement? Explain with the help of examples.

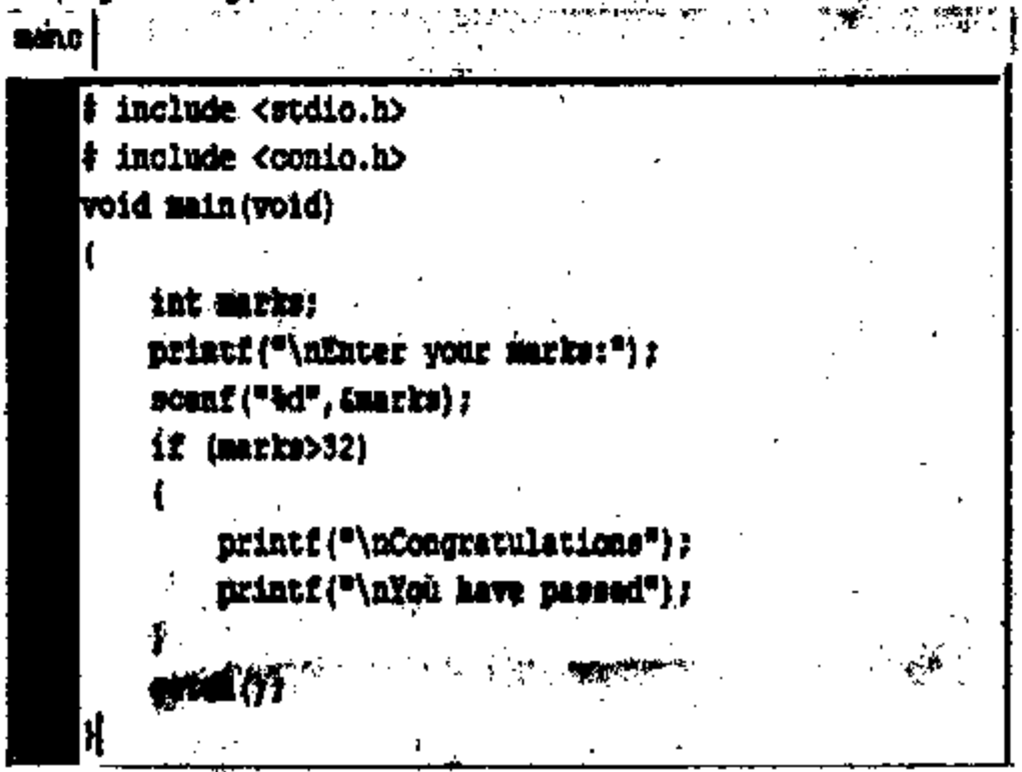
```
{
        Block of statements
}
```

When this statement is executed, the condition is evaluated. If the condition is true then the block of statements within the braces will be executed. If the condition is false then the block of statements within the braces will be skipped and the control will be transferred to the next statement if any exists.

If there is only one statement to be executed if the condition is true then braces are not required.

Use of If Statement (Examples):

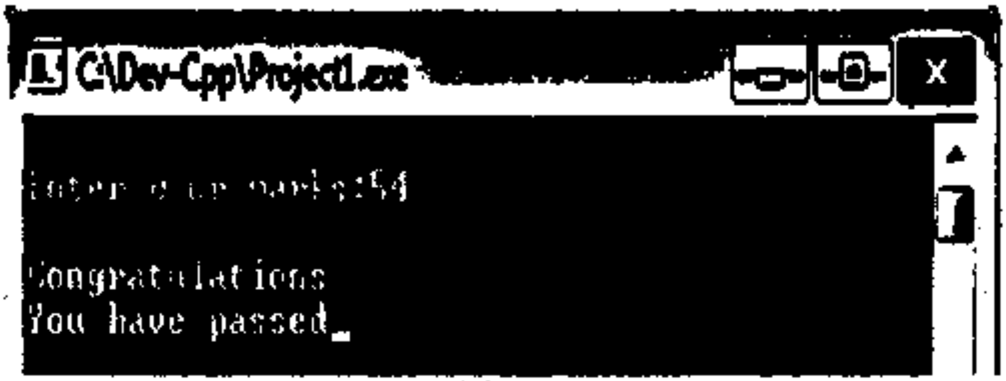
Program 1: The program in Fig., demonstrates the use of If statement.



```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int marks;
    printf("\nEnter your marks:");
    scanf("%d",&marks);
    if (marks>32)
    {
        printf("\nCongratulations");
        printf("\nYou have passed");
    }
    getch();
}
```

Program to demonstrate the use of if statement

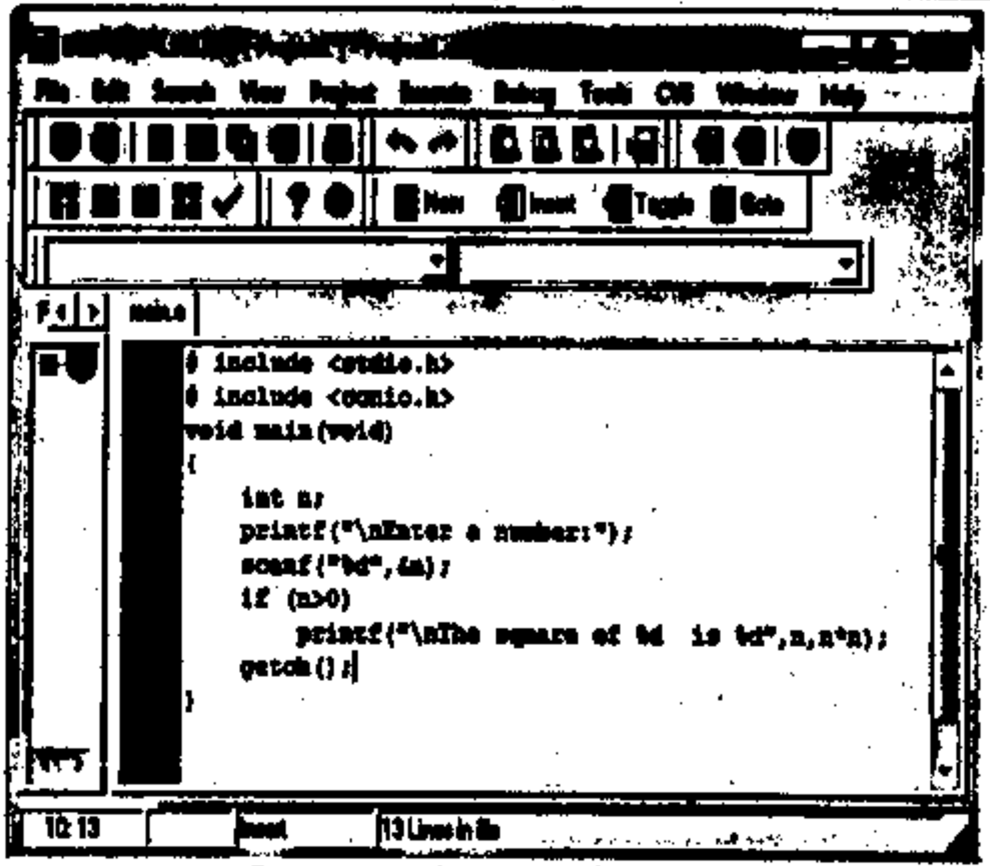
- When If statement of this program is executed, the condition inside the brackets is evaluated.
- If the condition is true, that is, the marks entered are greater then 32, then the two statements following the keyword If are executed.
- If the condition is false, that is, the marks entered are less than 33, then the following two statements will be skipped and the program will terminate and there will be no output.
- The output of the program is shown in Fig., if the marks entered are more than 32.



```
Enter your marks:54

Congratulations
You have passed_
```

Output of the program 1

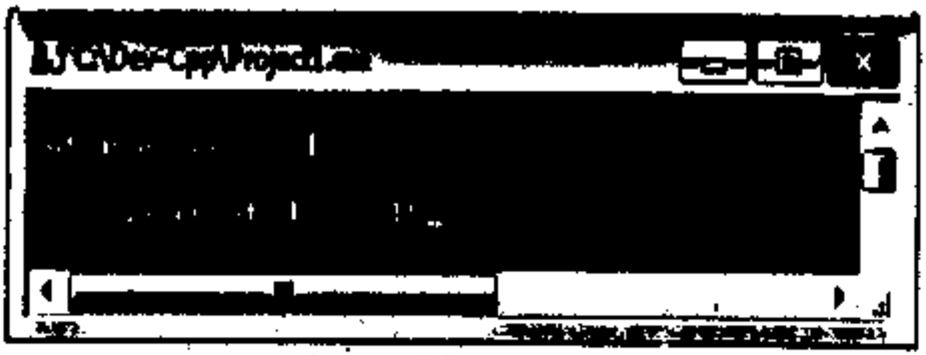
Program 2: The program in Fig., prints square of a number.


File Edit Search View Project Execute Debug Tools CVS Window Help
New
Insert
Toggle
Goto
main
10: 13
Insert
13 Lines in file

```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int n;
    printf("\nEnter a number:");
    scanf("%d",&n);
    if (n>0)
        printf("\nThe square of %d  is %d",n,n*n);
    getch();
}
```

Program to print square of a number

- When the above program is executed, it will prompt the user to enter a number.
- If the user enters a number greater than zero, it will print tbe square of the number as shown in Fig.



Output of the program 2

xiii. What is the purpose and structure of if-else statement? Explain with the help of examples.

Ans: Structure of If-else Statement:

The **if-else** statement is used in situation where some code is to be executed if a condition is true and some other code is to be executed if the condition is false.

The **if-else** statement has the following **general form / Syntax.**

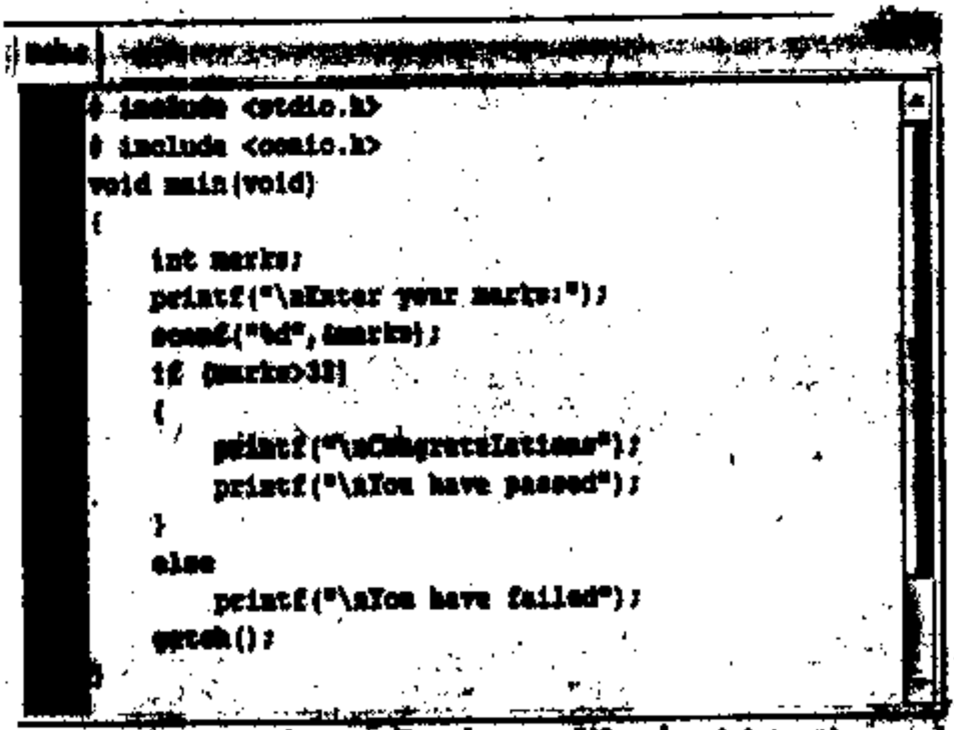
```
If (condition)
{
      Block of statements
}
else
{
      Block of statements
}
```

- When **if-else** statement is executed, the condition is evaluated.
- If the condition is true then the block of statements following If will be

- If the condition is false then the block of statements following if will be skipped and the block of statements following else will be executed.
- If a single statement is to be executed after if or else then braces are not required.

Use of If-Else Statement (Examples):

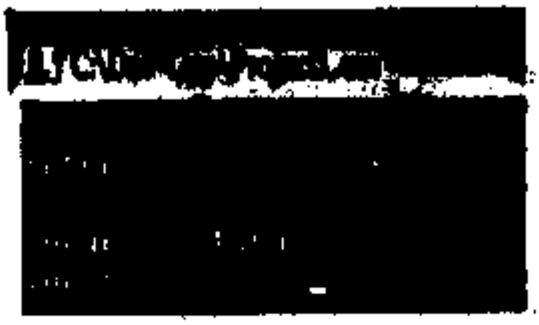
Program 1: The program in Fig., reads marks and prints the message whether the student passed or failed.

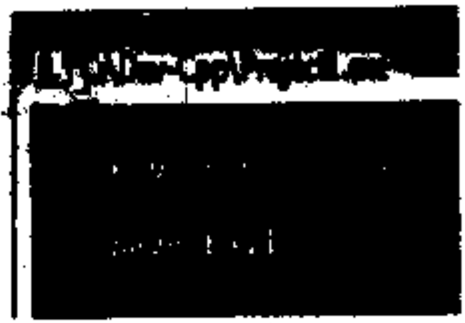


```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int marks;
    printf("\nEnter your marks:");
    scanf("%d",&marks);
    if (marks>32)
    {
        printf("\nCongratulations");
        printf("\nYou have passed");
    }
    else
        printf("\nYou have failed");
    getch();
}
```

Program to demonstrate the use of if – else statement

- In this program, if the marks entered are above 32 then the statements following if are executed and the message shown in Fig. (a) will be printed.
- If the marks entered are below 33 then the statement following else are executed and the message shown in Fig.(b) will be printed.





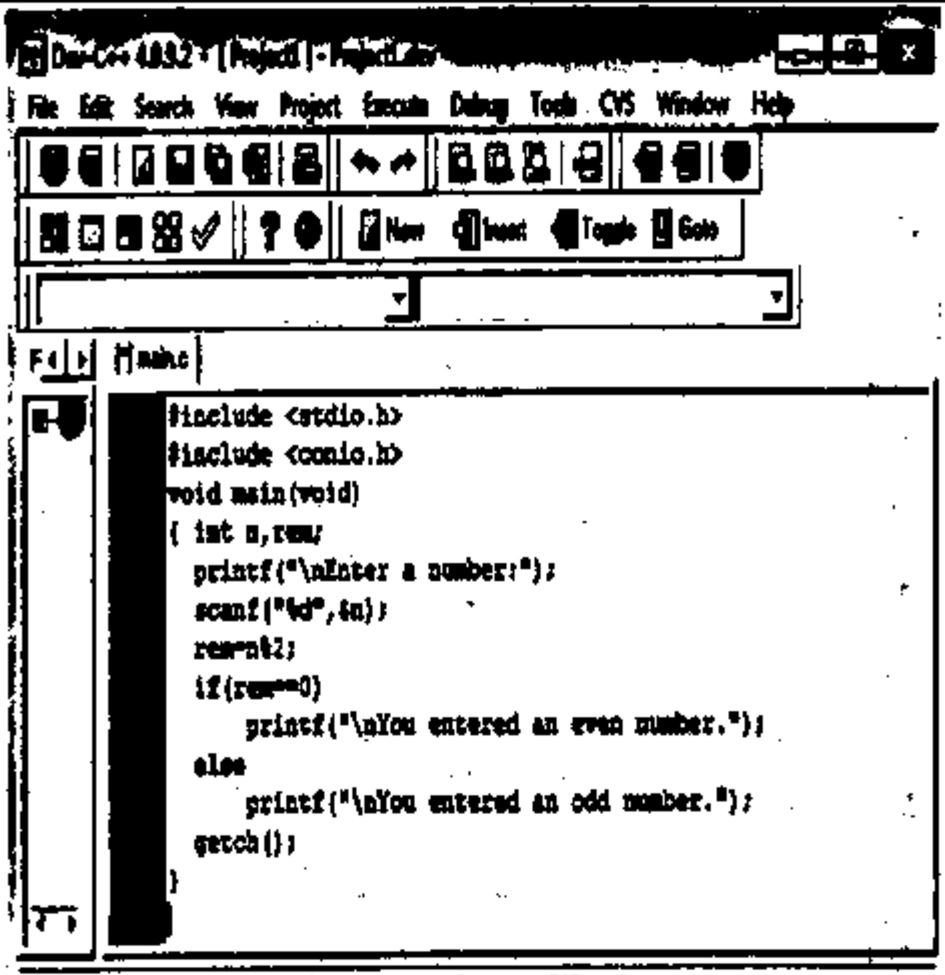
(a) Output when marks are grater than 32

(b) Output when marks are less than33

- Also note that, there is a single statement to be executed if the condition is false. Therefore, braces are not required after else.

Program 2:

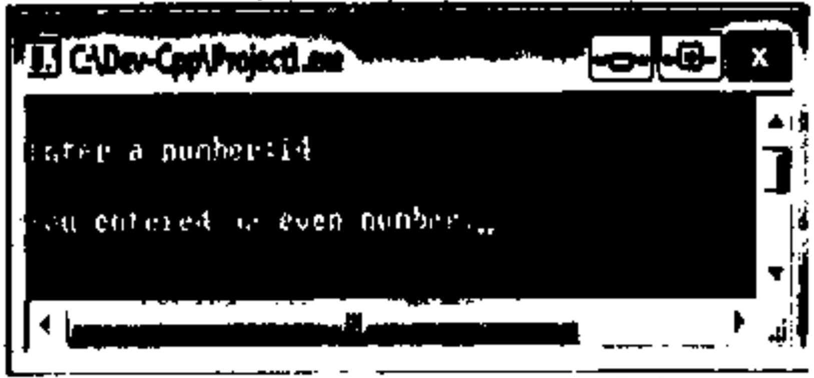
The program in Figure., reads a number from the keyboard and prints the message whether it is an even or an odd number.


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include <stdio.h>
#include <conio.h>
void main(void)
{ int n,rem;
  printf("\nEnter a number:");
  scanf("%d",&n);
  rem=n%2;
  if(rem==0)
      printf("\nYou entered an even number.");
  else
      printf("\nYou entered an odd number.");
  getch();
}
```

Program to demonstrate the use of if – else statement

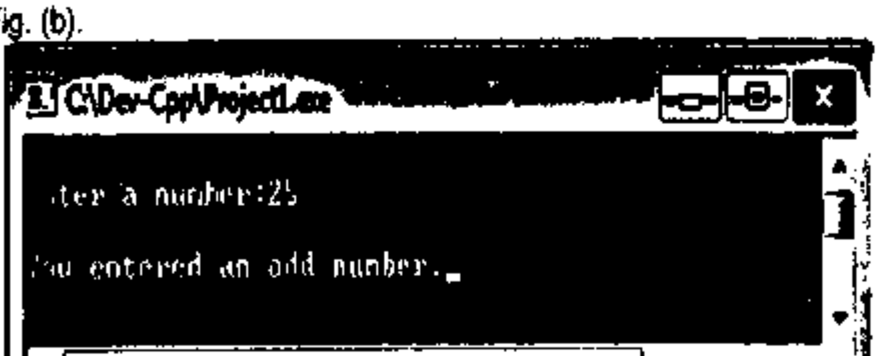
- When this program is executed it prompts the user to enter the value of **n.**
- The remainder operator % is used to divide the number by 2 and store the remainder in variable **rem.**
- The value of **rem** is checked. If it is equal to 0 then the number n, is an even number as shown in Fig. (a).



```
nter a number:14

ou entered an even number._
```

(a) Execution of program when value of n is 14.

- If the value of **rem** is not equal to 0, in other words, it is equal to 1 then the number **n** is an odd number as shown in Fig. (b).



```
ter a number:25

ou entered an odd number._
```

SECTION- C

Note: Attempt any TWO questions. All questions carry equal marks. (2 × 8 = 16)

Q3. What is control structure? Explain conditional control structure with examples.

Ans: Control Structure:

In a programming language, a control statement is an instruction which determines the sequence of execution of other statements in a program.

Control structures are used in programs to implement decisions.

Conditional control structure:

Conditional Statement:

A conditional statement is an instruction in a programming language that contains a condition. When a conditional statement is executed, first the condition is evaluated and then based on the result (true or false), a particular statement or a set of statements is executed.

Conditional statements of C language are **If, If-else, else-if** and **switch** statements.

Structure of If Statement:

The **If** statement has the following general form.

If (condition)

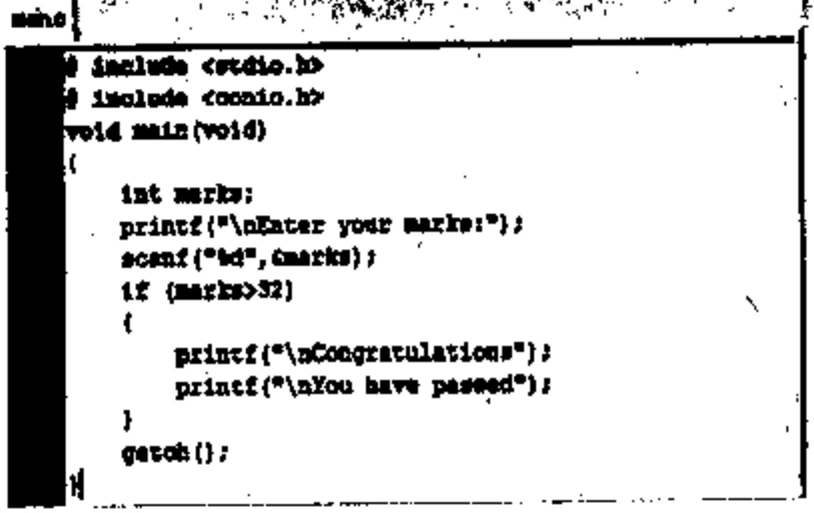
{

Block of statements

}

Use of If Statement:

Example: Program: The program in Fig., demonstrates the use of **If** statement.



```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int marks;
    printf("\nEnter your marks:");
    scanf("%d",&marks);
    if (marks>32)
    {
        printf("\nCongratulations");
        printf("\nYou have passed");
    }
    getch();
}
```

Program to demonstrate the use of If statement

Output of the program

Structure Of If-Else Statement:

The **If-else** statement is used in situation where some code is to be executed if a condition is true and some other code is to be executed if the condition is false.

The **If-else** statement has the following general form.

If (condition)

{

```
    {
        Block of statements
    }
```

- When **If-else** statement is executed, the condition is evaluated.
- If the condition is true then the block of statements following **If** will be executed and the block of statements following **else** will be skipped.
- If the condition is false then the block of statements following **If** will be skipped and the block of statements following **else** will be executed.
- If a single statement is to be executed after **If** or **else** then braces are not required.

Structure Of If-Else-If Statement:

The else-If is a type of conditional statement that combines more than two conditions. It allows the programmer to make a decision based on several conditions.

The **else-If** statement has the following general form.

```
  If(condition-1)
    {
        Block of statements
    }
    else If(condition-2)
    {
        Block of statements
    }
    else If(condition-3)
    {
        Block of statements
    }
            .
            .
            .
    else
    {
        Block of statements to be executed
        when none of the conditions is true.
    }
```

- When this statement is executed, condition-1 is evaluated, if it is true then the block of statements following **If** is executed and if it is false, the next condition is evaluated.
- If any condition is true then the following block of statements is executed.
- If none of the conditions is true then the block of statements following **else** is executed automatically.
- If a single statement is to be executed after **If**, **else If** or **else**, instead of a set of statements then the braces are not required.

Switch Statement:

The **switch** statement has the following general form.

```
        switch (expression)
        {
        case const-1:
            statements;
            break;
        case const-2:
                .
                .
        default:
                statements;
        }
```

- The **switch** statement is similar to the **else-if** statement. It is used when multiple choices are given and one choice is to be selected.

Q4. What is the purpose of switch () statement? Explain with the help of one example.

Ans: Switch Statement:

The **switch** statement has the following general form.

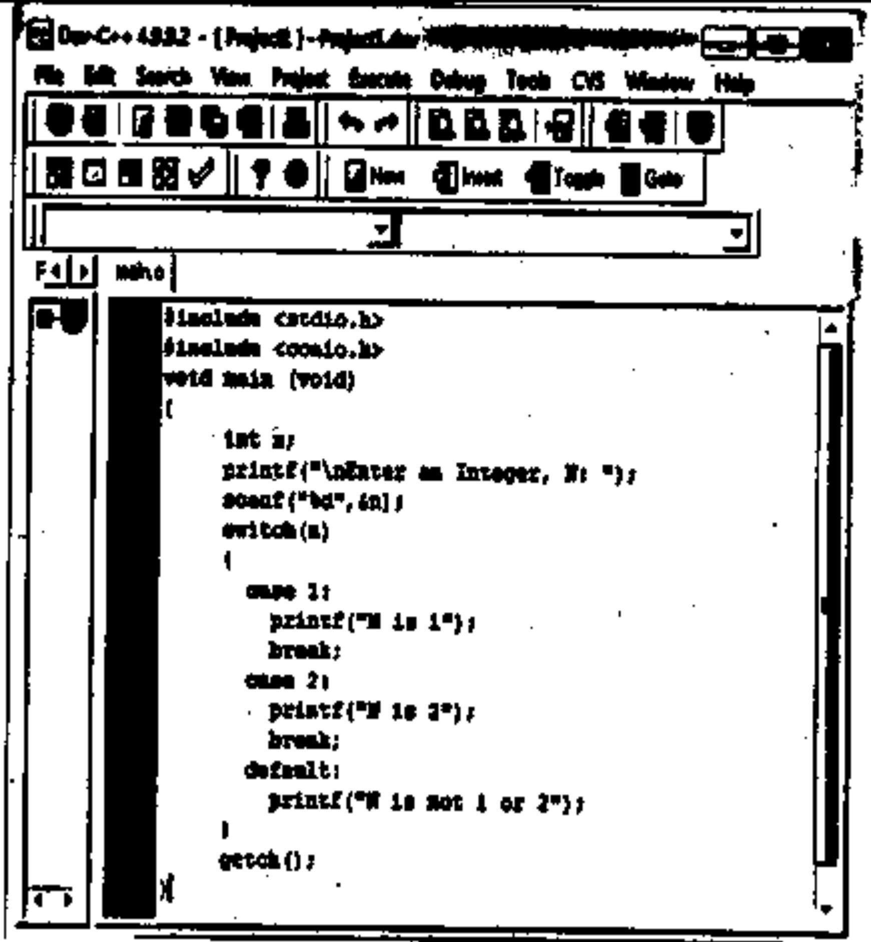
```
        switch (expression)
        {
        case const-1:
                statements;
                break;
        case const-2:
                statements;
                break;
                .
                .
                .
        default:
                statements;
        }
```

Purpose of switch statement:

- The **switch** statement is similar to the **else-if** statement. It is used when multiple choices are given and one choice is to be selected.
- When **switch** statement is executed, the expression is evaluated. Based on the result of expression one of the cases in the **switch** statement is executed. The result of expression is compared with the constant values given after the key word **case**. If the result matches the constant value after any **case** then the statements under that case are executed.
- In switch statement, it is allowed to use a variable within the parenthesis instead of an expression based on which statements under a **case** can be executed.
- The purpose of **break** statement is to exit the body of the **switch** statement after executing the statements under a **case** and transfer control to the first statement following the end of the **switch** statement.
- If no **case** is matched then the statements under the **default** keyword are executed. Its use is optional. If it is not used then the control exits from the body of the **switch** statement and goes to the first statement following the end of the **switch** statement.
- The expression should be of type **int, char** but not **float.**

Program:

The program in Figure., uses a variable of type integer as switch variable.


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c

```
#include <stdio.h>
#include <conio.h>
void main (void)
{
      int n;
      printf("\nEnter an Integer, N: ");
      scanf("%d",&n);
      switch(n)
      {
        case 1:
          printf("N is 1");
          break;
        case 2:
          printf("N is 2");
          break;
        default:
          printf("N is not 1 or 2");
      }
      getch();
}
```

Program to demonstrate use of switch statement

- When this program is executed, the switch variable must have an integer value. The value of switch variable n is compared with the constant values following the **case** keyword and if it matches, control is transferred to the statements following that particular **case**.
- If the switch variable does not match any of the **case** constants, control goes to the keyword **default** which is at the end of the switch statement.
- Notice the use of **break** statement in this program. It terminates the **switch** statement when the body of the statements in a particular **case** has been executed.

Q5. What is the purpose and structure of if-else-if statement? Explain with the help of examples.

Ans: Structure of If-Else-If Statement:

The **else-if** is a type of conditional statement that combines more than two conditions. It allows the programmer to make a decision based on several conditions.

The **else-if** statement has the following **general form / Syntax.**

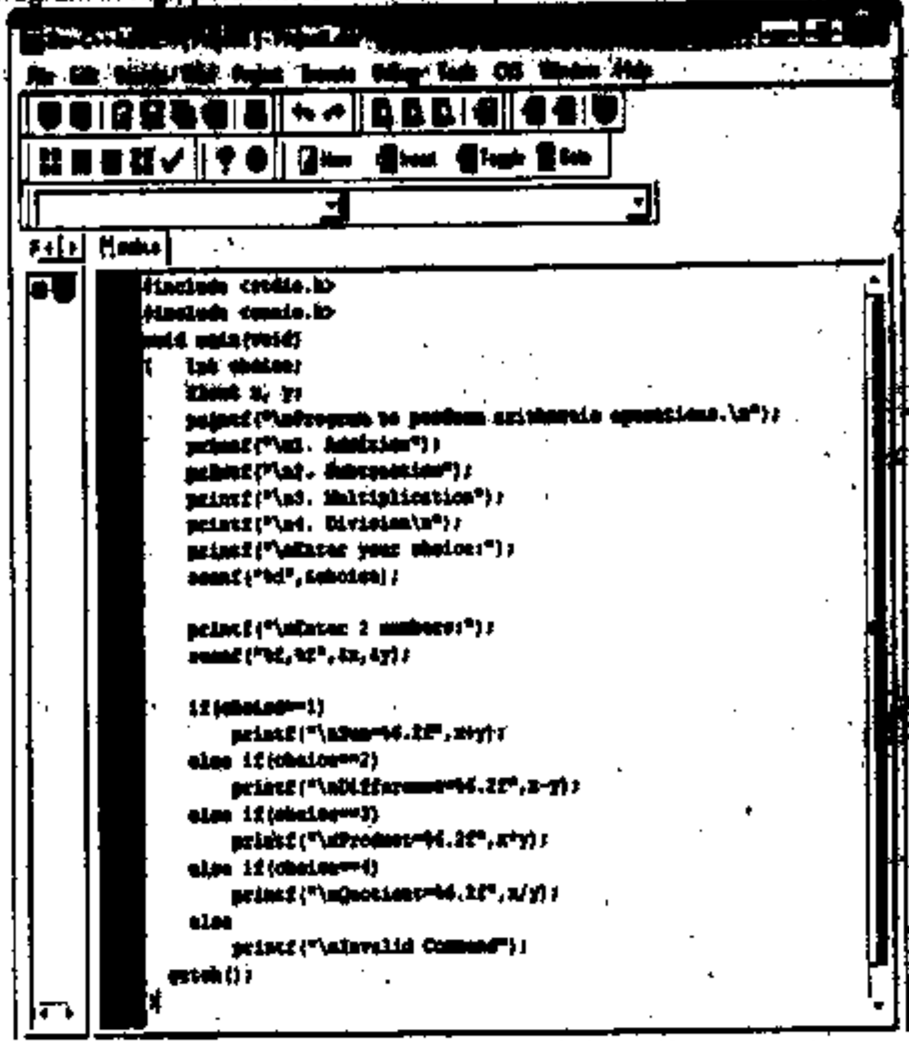
```
if(condition-1)
{
      Block of statements
}
else if(condition-2)
{
      Block of statements
}
else if(condition-3)
{
      Block of statements
```

```
else
{
        Block of statements to be executed
        when none of the conditions is true.
}
```

- When this statement is executed, condition-1 is evaluated, if it is true then the block of statements following if is executed and if it is false, the next condition is evaluated.
- If any condition is true then the following block of statements is executed.
- If none of the conditions is true then the block of statements following **else** is executed automatically.
- If a single statement is to be executed after **if, else if** or **else,** instead of a set of statements then the braces are not required.

Use of If-Else-If Statement (Examples):

Program 1: The program in Fig., performs an arithmetic operation based on the choice of user.



```
#include <stdio.h>
#include <conio.h>
void main(void)
{   int choice;
    float x, y;
    printf("\nProgram to perform arithmetic operations.\n");
    printf("\n1. Addition");
    printf("\n2. Subtraction");
    printf("\n3. Multiplication");
    printf("\n4. Division\n");
    printf("\nEnter your choice:");
    scanf("%d",&choice);

    printf("\nEnter 2 numbers:");
    scanf("%f,%f",&x,&y);

    if(choice==1)
        printf("\nSum=%6.2f",x+y);
    else if(choice==2)
        printf("\nDifference=%6.2f",x-y);
    else if(choice==3)
        printf("\nProduct=%6.2f",x*y);
    else if(choice==4)
        printf("\nQuotient=%6.2f",x/y);
    else
        printf("\nInvalid Command");
  getch();
}
```

Program to demonstrate the use of else – if statement

- When this program is executed, it will display four choices and prompt the user to enter his choice. User's choice will be stored in the variable **choice.**
- After this, it will again prompt the user to enter two numbers which will be stored in variables **x** and **y.**
- Now, the **if-else-if** statement will be executed. First condition will be checked first, if it is true, that is, the value stored in the variable **choice** is **1** then sum of **x** and **y** will be evaluated and printed.
- If the first condition is false then the second condition will be checked and if it is true, then the difference **(x-y)** will be evaluated and printed and so on.
- If all the conditions are false then the statement following else will the executed and the message **"Invalid Command!"** will be printed.